2013

DISEÑO Y DESARROLLO DE UNA APLICACIÓN CLIENTE-SERVIDOR PARA LA GESTIÓN DE GRUPOS DE TRABAJO EN UN ENTORNO EDUCATIVO COLABORATIVO ORIENTADO A DISPOSITIVOS ANDROID

GRADO EN INGENIERIA DE SISTEMAS AUDIOVISUALES

TRABAJO DE FIN DE GRADO

Tutor: Iria Manuela Estévez Ayres
Alumno: David Argüeso González

Universidad
Carlos III de Madrid

AGRADECIMIENTOS

En primer lugar quiero dar las gracias a mi padre, a mi madre y a mi tía, que han estado a mi lado apoyándome en todo momento a lo largo de estos años. Ellos son los que me han ayudado en este atareado camino y me han dado fuerzas para conseguir todo lo que me proponga.

A toda mi familia, por el interés y el apoyo recibido durante este periodo, con especial afecto a mis abuelos y abuelas, a los que me hubiese gustado decir: "Ya he acabado la carrera".

A Andrea, que ha estado en todo momento dándome ánimos para seguir adelante, ayudándome a levantarme en cada caída.

A Juan Manuel Pozo Santiago, mi fiel compañero durante la carrera. Espero que no se pierda la relación ya que es un gran amigo.

A mi tutora de proyecto, Iria Manuela Estévez Ayres, por ser una gran profesora y por brindarme la oportunidad de realizar este trabajo.

Y como no, a todos mis compañeros de universidad, con los que he pasado grandes momentos.

Para terminar, dar las gracias a cualquiera que se preocupase por como me iba en la carrera durante estos años.

Gracias.

David

Junio 2013

Universidad
Carlos III de Madrid

"La posibilidad de realizar un sueño es lo que hace que la vida sea interesante."

Paulo Coelho

Universidad
Carlos III de Madrid

RESUMEN

Este proyecto consiste en la implementación de una aplicación para teléfonos inteligentes basados en el sistema operativo Android de Google. Para ello se realiza un estudio detallado de las diferentes tecnologías disponibles, para posteriormente escoger las más adecuadas para el desarrollo de cada bloque del proyecto.

Se propone el desarrollo de una aplicación que gestione los grupos de trabajo de las distintas asignaturas de la Universidad Carlos III de Madrid, facilitando a los alumnos la organización diaria y la comunicación entre ellos.

La ejecución de este proyecto se divide en tres bloques principales.

El primero, la implementación de una interfaz sencilla y de fácil manejo de una aplicación móvil. El segundo es la creación de un servidor al que el cliente realizará las consultas para que le proporcione los datos necesarios. Por último se desarrolla una base de datos donde se guardarán los registros de todos los grupos de trabajo de la universidad.

Para finalizar el proyecto, se han realizado una serie de pruebas que verifican el correcto funcionamiento de la aplicación, mostrando capturas de pantalla de algunas secciones en este documento.

Universidad Carlos III de Madrid

ÍNDICE

Universidad Carlos III de Madrid

ÍNDICE DE ILUSTRACIONES

Universidad Carlos III de Madrid

ÍNDICE DE TABLAS

GLOSARIO DE ACRÓNIMOS

1.	AGPL	Affero General Public License
2.	APK	Application Package File
3.	CGI	Common Gateway Interface
4.	GB	GigaByte
5.	DTD	Document Type Definition
6.	GHZ	Giga HertZ
7.	GML	Generalized Markup Language
8.	GNU GPL	GNU is not UNIX General Public License
9.	GPS	Global Positioning System
10.	HTTP	HyperText Trensfer Protocol
11.	IOS	IPhone Operating System
12.	IP	Internet Protocol
13.	JSON	JavaScript Object Notation
14.	MB	MegaByte
15.	MHZ	Mega HertZ
16.	PDF	Portable Document Format
17.	RAM	Random Access Memory
18.	RIM	Research In Motion
19.	ROM	Read-Only Memory
20.	SGML	Standard Generalized Markup Language
21.	SO	Sistema Operativo
22.	SQL	Structured Query Language
23.	TB	TeraByte
24.	URL	Uniform Resource locator
25.	USD	United States Dollar

CAPÍTULO 1. INTRODUCCIÓN

La implantación del Plan Bolonia en las universidades ha supuesto un cambio en la forma de trabajar del alumno, que se ve obligado a llevar las asignaturas al día, debido a los exámenes parciales, trabajos y prácticas individuales o en grupo que se realizan durante el curso.

El principal objetivo de este proyecto es el desarrollo de una aplicación que favorezca la gestión de grupos de prácticas de las asignaturas de la Universidad Carlos III de Madrid. Para ello se implementarán una serie de aplicaciones mediante las que los alumnos puedan estar en contacto y organizar su tiempo a lo largo del cuatrimestre.

Para facilitar el acceso y la comunicación de los alumnos, se decide implementar una aplicación que se ejecute sobre un teléfono móvil. De esta manera, los alumnos podrán tener acceso a la aplicación desde cualquier lugar, siempre que dispongan de conexión a internet, y poder tener organizados sus grupos de trabajo en todo momento.

1.1 SMARTPHONES

La evolución de la tecnología móvil ha dado un gran salto en los últimos años. En su inicio, los teléfonos móviles estaban pensados para realizar y recibir llamadas, pero debido a su gran evolución, se han convertido en teléfonos inteligentes capaces de realizar múltiples funciones como GPS (Global Positioning System [9]), cámara de fotos y vídeo, internet o reproductor multimedia, denominándose ahora smartphones.

Los primeros smartphones salieron al mercado en la década de los noventa, pero estaban destinados a altos ejecutivos debido a su precio. Posteriormente empezaron a aparecer dispositivos de gama media más económicos que propiciaron el auge de estos terminales en el mundo hasta el día de hoy. No todos los dispositivos disponen de las mismas características ni las mismas funciones, de ahí esa variabilidad en los precios.

Estas funciones quedarían desaprovechadas de no ser por los desarrolladores de cada SO (Sistema Operativo [21]) que implementen programas que utilicen las diversas funciones del teléfono. Estas aplicaciones se pueden encontrar en las diferentes tiendas de software como 'Google Play' o 'App Store' de donde podrán descargarse gratuitamente o previo pago.

1.2 SISTEMAS OPERATIVOS

Hay diferentes sistemas operativos en los que funcionan estos dispositivos. Entre los más importantes podemos citar: Android (Google), iOS (IPhone Operating System [11], Apple), RIM (Research In Motion [18], BlackBerry), Symbian (Nokia) y Windows Phone (Microsoft).

El sistema operativo Android está cobrando cada vez más importancia en el mercado de los smartphones, debido a que es un sistema operativo libre y gratuito y esto hace que los fabricantes puedan desarrollar sus propias ROMs (Read-Only Memory [19]) de forma que puedan optimizar las funciones de cada dispositivo y hacerlos fácilmente manejables para el usuario.

1.3 ORGANIZACIÓN DE LA MEMORIA

El presente documento se organiza de la siguiente manera:

- Capítulo 2. Consiste en una explicación detallada de las posibles tecnologías que se pueden utilizar para las diferentes partes del proyecto.
- Capítulo 3. En este capítulo, se detalla el diseño de las diferentes partes del proyecto.
- Capítulo 4. Contiene los medios utilizados para la realización del proyecto así como las diferentes fases de implementación que se han seguido a lo largo del desarrollo del proyecto, los programas para su implementación y la explicación detallada de cada bloque de la aplicación.
- Capítulo 5. En éste se enumerarán los dispositivos con los que se han realizado las pruebas y se mostrarán las capturas de pantalla de la aplicación.
- Capítulo 6. Aquí se exponen las conclusiones obtenidas al concluir el desarrollo del proyecto.
- Capítulo 7. Comprende una planificación temporal de la aplicación y un presupuesto del coste económico del mismo.
- Capítulo 8. Para finalizar se incluye la bibliografía donde se incluyen las páginas web que se han consultado para el desarrollo de la memoria y de la aplicación.

CAPÍTULO 2. ESTADO DEL ARTE

En este capítulo se realiza una comparación de las diferentes tecnologías que se pueden aplicar para el desarrollo de cada bloque de la aplicación, así como una decisión final de la tecnología escogida.

2.1 MOTIVACIÓN

Los grandes avances de la tecnología suponen un aumento en la aparición de dispositivos inteligentes. Cualquiera que tenga un Smartphone que tenga conexión a internet dispone de una gran oferta de aplicaciones con multitud de servicios y funcionalidades donde escoger.

Estas aplicaciones son utilizadas por una gran parte de los propietarios de un dispositivo con estas características. En una universidad donde la gran mayoría de personas son jóvenes, el porcentaje de posesión de estos terminales es muy elevado a causa de la necesidad de estar actualizados tecnológicamente de este colectivo de la sociedad.

Por lo que se ha encontrado la necesidad de crear una aplicación para smartphones, en la que los alumnos se puedan comunicar para la realización de estos trabajos y prácticas en grupo

La aplicación tiene un objetivo docente ya que con ella pretendemos solucionar los problemas de comunicación por parte de los alumnos, dado que estos grupos se pueden formar aleatoriamente por el profesor de forma que esté compuesto por compañeros que no tengan mucha relación en clase, o incluso que pertenezcan a grupos diferentes y no coincidan durante el horario lectivo.

2.2 OBJETIVOS

El principal objetivo de este proyecto es la implementación de una aplicación de fácil utilización e intuitiva, que ayude a los alumnos de la Universidad Carlos III de Madrid en su gestión de grupos de trabajo de sus diferentes asignaturas.

Para el desarrollo, se divide la aplicación móvil en varias secciones básicas que son: un chat y una gestión de eventos a través de un calendario. Para ello se han de desarrollar tres bloques generales que son: una aplicación para un smartphone, un servidor y una base de datos.

A continuación se muestra el esquema que sigue el proyecto a desarrollar en la *Ilustración 1*

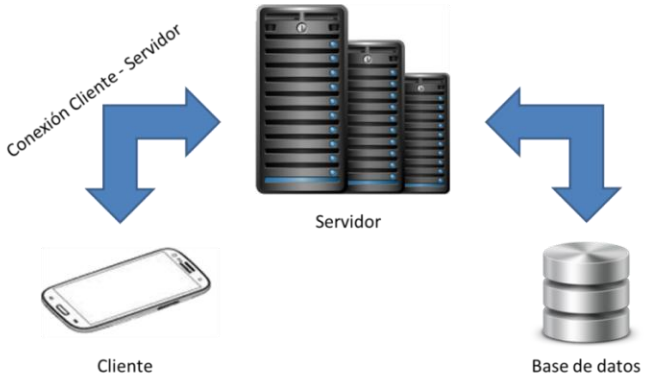


Ilustración 1. Esquema de conexión Cliente- Servidor

Para la realización de este trabajo, se ha tenido que realizar un estudio de las posibles opciones de las tecnologías para el desarrollo de cada parte de la estructura de la aplicación.

2.3 CLIENTE

En la actualidad existen diversos sistemas operativos entre los que escoger para desarrollar una aplicación móvil: Android (Google), iOS (Apple), Windows Phone (Microsoft), Symbian (Nokia), RIM (BlackBerry), Bada, Meego, Maemo, etc.

De todos ellos, los dominantes del mercado son Android e iOS y como la idea de este proyecto es implementar una aplicación que sea utilizada por la gran mayoría del alumnado de la universidad, se escogerá uno de estos dos sistemas y por lo tanto se hará un análisis más detallado de cada uno, justificando el porqué de esa elección.

2.3.1 ANDROID

Android[1] es un SO libre perteneciente a la empresa de Google basado en Linux, por lo que es una gran ventaja a la hora de desarrollar aplicaciones o mejorar el sistema sin depender del fabricante ni de la operadora. Esta libertad de desarrollo permite implantar esta plataforma en otros dispositivos de nuestra vida cotidiana como GPS, televisores, ordenadores…

Permite un amplio rango de posibilidades para personalizar el teléfono tanto visualmente (widgets, o fondos de escritorio) como funcionalmente (a través de aplicaciones). La oferta de dispositivos sobre la plataforma Android es muy amplia, debido a la amplia variedad de terminales en el mercado.

La oferta de aplicaciones en el Google Play no es tan amplia como la del App Store, pero una ventaja de Android es que como no tiene un control tan estricto, permite instalar aplicaciones de orígenes desconocidos a través de un archivo con extensión .apk (Application Package File $_2$) y por lo tanto no depender del servicio de Google para instalar nuevas aplicaciones.

Gracias al sistema multitarea es capaz de mantener varias aplicaciones abiertas a la vez, pudiendo gestionarlas de forma que las suspenda o las cierre si no se están utilizando para ahorrar batería. Pero como contraprestación, esto acarrea un mayor consumo de la batería dado que el mantener las aplicaciones en segundo plano supone un mayor gasto.

Las baterías de un smartphone se agotan rápidamente debido a la conexión a internet y la constante sincronización de datos. Las baterías de la mayoría de los terminales con Android, a diferencia de su gran rival (iOS), son extraíbles y se puede comprar una de repuesto para sustituirla en caso de necesidad.

Dada la variedad de características de los terminales, muchos disponen de un slot para poder ampliar la memoria interna del teléfono mediante una tarjeta de memoria. Esto ayuda mucho para los dispositivos en los que la memoria interna del teléfono es muy reducida, mientras que en los dispositivos de alta gama es un suplemento de memoria en el que el usuario puede complementar la memoria interna.

Sin embargo, la fragmentación del sistema de Android supone un descontento entre los usuarios ya que al existir tantas versiones, los teléfonos se van quedando desactualizados[1]. Periódicamente los fabricantes pueden publicar actualizaciones de una versión que suponen una mejora de funcionamiento o arreglos de bugs que han ido surgiendo, pero no tiene por qué cambiar la versión del SO a una más reciente, y esto causa problemas con la compatibilidad de aplicaciones.

Dado que el sistema es libre, la seguridad de estos dispositivos es escasa y por lo tanto el acceso al sistema por parte de software indeseado es inevitable. Una solución a este problema es instalar aplicaciones seguras y fiables y un antivirus para así evitar este tipo de software malicioso

2.3.2 IOS

IOS[2] es un SO cerrado perteneciente a la empresa de Apple. Es un sistema que tiene un control severo del software y hardware de manera que la integración de ambos está muy bien conseguida.

A la hora de desarrollar programas para esta plataforma es necesario disponer de equipos de la marca Apple, lo que limita la implementación de nuevas aplicaciones debido al elevado precio de los ordenadores de la marca.

La personalización de estos terminales es escasa por no decir que es casi nula, a causa de este estricto control para asegurar la estabilidad de los dispositivos.

Las aplicaciones solo pueden obtenerse a través de la App Store[3] que tiene un riguroso control de las aplicaciones, pero a pesar de ello cuenta con más aplicaciones que Google Play[4].

El precio de estos dispositivos en el mercado es bastante elevado y la variedad solo difiere en el tamaño de la memoria interna del teléfono, el color y forma del terminal.

Los terminales siempre están actualizados a la última versión del SO gracias a las actualizaciones automáticas realizadas cuando se conecta el dispositivo a un ordenador y al programa correspondiente.

Estos dispositivos no disponen de Adobe Flash Player por lo tanto la visualización de páginas web que contengan objetos flash no será posible.

La transferencia de archivos con otros dispositivos que no tengan un SO iOS es inviable debido a que, esta opción está limitada solo para terminales iOS.

Después de este pequeño estudio se decide realizar el proyecto en Android, principalmente por la disposición de medios para el desarrollo y la imposición por parte de la tutora, además de que es más llamativo un sistema en el que no se tiene ningún tipo de restricciones en el sistema, donde se puede experimentar con todas las opciones que da el terminal. Destacar que la aplicación se va a desarrollar para dos sistemas operativos, de forma que dos alumnos diferentes implementan dos aplicaciones móviles, y comparten la implementación del servidor y de la base de datos.

2.3.3 ESTRUCTURA ANDROID

En la *Ilustración 2* se ve la organización de carpetas de un proyecto en Android.

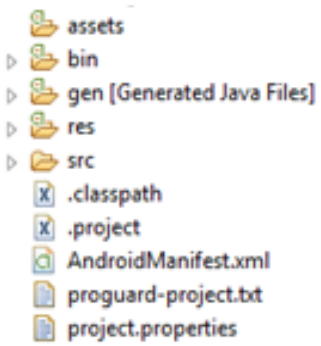


Ilustración 2. Estructura de un proyecto Android

- /assets: En ella se encuentran los ficheros necesarios para la aplicación como los ficheros de configuración, datos etc. A diferencia de los archivos de /res/, a estos ficheros no se les generan un ID en la clase R.java para luego poder recuperarlos si no que se tiene que especificar la ruta del fichero al leerlo y usar la clase AssetManager para ello.
- /bin: Contiene los ficheros compilados de la aplicación y otros elementos auxiliares. Hay que señalar que es en esta carpeta donde se encuentra el archivo .apk de la aplicación, que es el archivo de instalación de la aplicación en un terminal.
- /gen: Engloba unos ficheros generados automáticamente cuando se compila el proyecto, que no se pueden modificar ya que contienen los códigos necesarios para la gestión de los recursos de la aplicación.
- /res: En esta carpeta se meten todos los recursos utilizados en el proyecto como los diseños de las diferentes pantallas de la interfaz, imágenes, definición de cadenas de texto y colores, etc. Todo esto se divide en subcarpetas como se puede ver en la *Ilustración 3*.

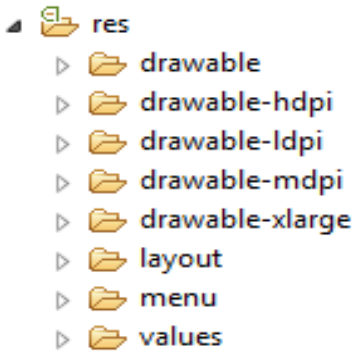


Ilustración 3. Esquema de la carpeta /res

- /src: Contiene todo el código fuente de la aplicación agrupado en paquetes.

En caso de necesitar incluir una librería en nuestro proyecto se genera una nueva carpeta llamada libs en la que se introducen las librerías en formato .jar.

Se observa también en la *Ilustración 2* un archivo denominado AndroidManifest.xml que es el que contiene la definición de los parámetros principales de la aplicación como el nombre, la versión, los permisos, las pantallas que se mostraran, las librerías, etc.

2.3.1.1. Activities

Como muestra la documentación de Google, una activity es "es una cosa única con un objetivo determinado que el usuario puede hacer"[5]. Si se intenta buscar una definición aplicada a la práctica, se puede decir que una actividad es una clase java que hereda de la clase Activity.java, que mostrará al usuario una ventana en la que se mostrará la interfaz diseñada por el programador. Todas las actividades tienen que estar declaradas en el archivo AndroidManifest.xml.

Se organizan como una pila de objetos, es decir, al iniciar una nueva actividad la anterior queda por debajo, de forma que cuando la nueva actividad finalice, se vuelve a mostrar la que estaba debajo de la pila.

En la *Ilustración 4* se muestra el ciclo de vida de una actividad, es decir, los diferentes estados por los que pasa:

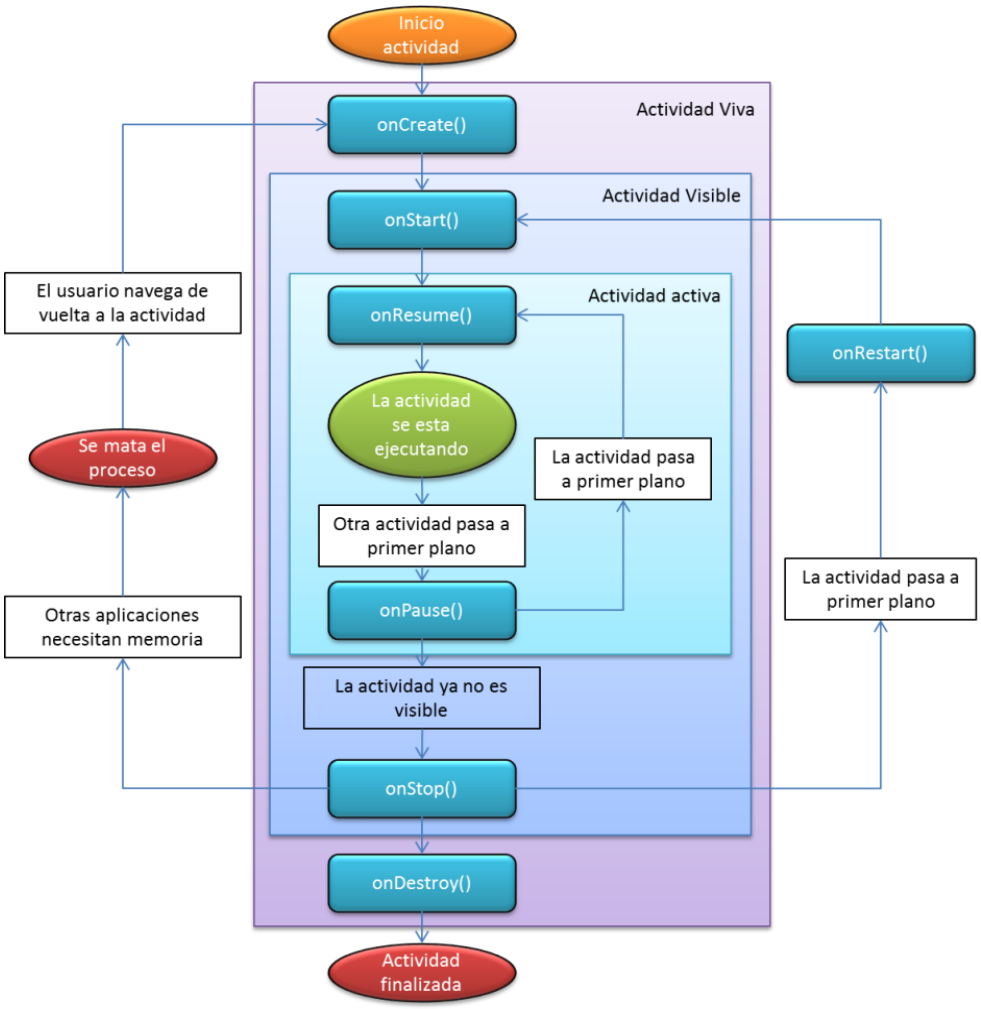


Ilustración 4. Ciclo de vida de una actividad [5]

El ciclo de vida de una actividad abarca desde la llamada al método onCreate() hasta la llamada al método onDestroy(). Realiza la configuración del estado global en el onCreate y libera los recursos en el onDestroy.

El ciclo visible ocurre desde la llamada a onStart hasta el método onStop. Durante este intervalo de tiempo el usuario puede ver la pantalla aunque puede no estar en primer plano interactuando con él. Durante estos métodos los recursos necesarios para mostrar la actividad se mantienen. Pueden ser llamados múltiples veces.

El tiempo en el que una aplicación está en primer plano comprende los métodos onResume y onPause. Durante este tiempo la actividad está la primera de la pila de manera que es la que se está mostrando al usuario.

2.4 CONEXIÓN CLIENTE – SERVIDOR

El cliente realiza peticiones a un servidor que le pasara una serie de datos que el cliente necesita para continuar la ejecución. Para ello se tendrán que mandar entre ambos una serie de datos a través de la red. Esta información se puede enviar encapsulada en diferentes tipos de datos. A continuación se detallan algunos de ellos.

2.4.1. XML

El nombre de este formato proviene de 'eXtensible Markup Language'[6] o lenguaje de marcas extensible. Es un formato desarrollado por el W3C (World Wide Consortium), que deriva de SGML (Standard Generalized Markup Language [20]) que a su vez proviene de una normalización de GML (Generalized Markup Language [7]) y permite definir la gramática de lenguajes determinados para organizar y gestionar grandes documentos.

XML tiene extensión para diferentes aplicaciones, tanto para internet, como para el intercambio de datos en bases de datos editores de texto etc.

Mediante otras tecnologías que complementan a XML se consiguen unos desarrollos mucho mayores con más posibilidades haciendo de la transmisión de datos un medio fiable, seguro y fácil.

Algunas ventajas de XML:

- Más sencillo que SGML debido a las mejoras desarrolladas sobre éste.
- Es posible extender XML de forma que se puedan añadir nuevas etiquetas.
- No es necesario tener un analizador para cada versión de XML ya que el analizador es estándar y hace posible el desarrollo de elementos XML fácilmente.
- El código es fácil de entender por lo que cualquiera que lea un archivo XML no va a tener dificultad en su comprensión
- Se transforman los datos en información de manera que se tiene mayor flexibilidad a la hora de organizar los documentos.

Las partes en las que se compone un archivo XML son las siguientes:

- Prólogo: No es obligatorio ponerlo, es la parte donde se describe información del archivo como la versión utilizada de XML, el tipo de documento etc.
- Cuerpo: Es obligatorio y es necesario que tenga un solo elemento raíz al que se irán añadiendo los demás elementos.
- Elementos: Son cada propiedad del objeto que se define en el XML.
- Atributos: Son las propiedades o características que tienen cada uno de los elementos declarados dentro del cuerpo
- Entidades predefinidas: Son descripciones especiales para caracteres que el procesador XML pueda considerar como elementos al confundirse con palabras reservadas para el lenguaje XML.

- Secciones CDATA: Son partes del código en las que el procesador XML no va a comprobar los datos de forma que de error.
- Comentarios: Son líneas de código que se incluyen de modo explicativo para el programador pero que son invisibles a la hora de ser leídas por el procesador.

Para que un documento sea válido se comprueba la sintáctica básica, es decir, que esté bien formado el esquema del archivo (elementos, atributos etc). Para ello se comprueba un documento externo siendo a veces el DTD (Document Type Definition [5]) que es la Definición de Tipo de Documento, o el XSchema donde se realiza una definición de los elementos o atributos utilizados que no están incluidos en el lenguaje XML utilizado.

Se puede generar cualquier archivo con formato XML con cualquier procesador de texto, aunque existen algunos programas de desarrollo que facilitan la creación de estos ficheros ya que reconoce el formato y ayuda con la programación, incluyendo colores u organizando los diferentes elementos visualmente.

A continuación, en la *Ilustración 5* se muestra un ejemplo de un archivo XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Ejemplo SYSTEM " Ejemplo.dtd">

<Ejemplo>
     <Persona>
          <Id>p1</Id >
          <Nombre>David</Nombre>
          <Edad>24</ Edad >
          <Altura>1.95</ Altura >
          <Atributos>
               <Atributo1>Simpático</Atributo1>
               <Atributo1>Trabajador</Atributo2>
               <Atributo1>Alegre</Atributo3>
          </Atributos >
     </Persona >
</Ejemplo>
```

Ilustración 5. Ejemplo de Código XML

2.4.2. JSON

JSON[7] son las siglas de JavaScripst Object Notation [13] , que definen un formato de codificación para en envió y recepción de datos. El uso de JSON es más sencillo que el de XML dado que es más fácil hacer un analizador sintáctico. JSON se usa normalmente en aplicaciones donde se requiere un gran tamaño de datos tanto de envió como de recepción.

Es habitual ver en aplicaciones la utilización de ambos formatos a pesar de estar ambos enfrentados en cuanto a su uso.

Un ejemplo de definición de un objeto JSON es el mostrado en la *Ilustración 6*.

```
{"persona":
  {"id": "p1",
   "nombre": "David",
   "edad": “24”,
   "altura": “1.95”,
   "atributos":
        ["Simpático", "Trabajador", “Alegre”]
  }
}
```

Ilustración 6. Ejemplo de Código JSON

2.5 SERVIDOR

Un servidor[8] es un módulo que aporta servicios a una aplicación llamada cliente. Dependiendo del uso que se le vaya a dar al servidor se puede utilizar un equipo u otro, es decir para una gestión en la que se almacene mucha información o se necesite una gran capacidad de procesado, se necesitará un equipo de altas prestaciones que lo soporte, mientras que si el trabajo que se va a desarrollar en el servidor no es necesaria una gran máquina. Incluso se puede tener en el mismo equipo el cliente y el servidor realizando pequeñas conexiones para aplicaciones sencillas.

Existen varios tipos de servidores que se muestran a continuación:

- Servidor de impresión: Controla las impresiones de documentos de la red en la que está conectada, organizando la cola de trabajos a imprimir.
- Servidor de correo: Gestiona operaciones relacionadas con el correo electrónico.
- Servidor de fax: Organiza las operaciones que se pueden realizar con faxes.
- Servidor de telefonía: Gestiona las aplicaciones de telefonía como el contestador automático
- Servidor proxy: Realiza las operaciones que un cliente de la red le solicita para la gestión de la aplicación. Ofrece un servicio de cortafuegos, es decir, que aporta seguridad al sistema gestionando el acceso a internet dentro de una red.
- Servidor de acceso remoto: Gestiona el acceso a otros dispositivos localizados en una red remota.
- Servidor de uso: Organiza la parte lógica de la informática
- Servidor web: Realiza gestiones con todo tipo de documentos Web enviándolos a través de la red a los clientes que lo soliciten
- Servidor de base de datos: Ofrece operaciones con bases de datos a las aplicaciones que lo requieran.
- Servidor de reserva: Asegura la estabilidad de la red, de manera que si se pierde algún servidor principal, se recupere la información de manera que no haya pérdidas. Este modelo se conoce como clustering.

- Servidor de seguridad: Brinda protección al sistema, deteniendo accesos indeseados como virus, spyware o malware. Dependiendo de la cantidad de servidores de seguridad se aporta mayor o menor fiabilidad en el sistema.

Pero según su uso se dividen en dos grupos:

- Servidor dedicado: Gestionan únicamente las peticiones de la red, dedicando toda su funcionalidad a la red.
- Servidor no dedicado: Comparte la potencia entre los clientes y las operaciones locales que se puedan desarrollar dentro del equipo local.

Se decide por utilizar un servidor dedicado en el que se operen solo las gestiones realizas por un cliente externo. Para ello se estudian diferentes tecnologías que a continuación se detallan.

2.5.1. INTERFAZ DE ENTRADA COMÚN

Es una tecnología de World Wide Web que permite a través de un navegador web a un cliente hacer peticiones de datos a una aplicación ejecutándose en un servidor web. Esta interfaz es conocida como CGI (Common Gateway Interfaz $_3$)[9] y es un sistema de comunicación entre el servidor y el cliente.

Este tipo de aplicaciones fueron las primeras que utilizaban contenido dinámico para las páginas web, es decir, la información que se genera en el momento que es consultada. El servidor manda las peticiones que solicita el cliente a un servicio externo que a su vez le devuelve al cliente la información que ha solicitado.

Los pasos que se siguen en una comunicación de este tipo son los siguientes:

1. Primeramente, el servidor obtiene la petición del cliente y comprueba que se trata de una petición de un servicio CGI.
2. A continuación, el servidor organiza el entorno para ejecutar el servicio mediante la información que el cliente ha enviado.
3. Una vez listo, se ejecuta la aplicación obteniendo la salida estándar.
4. El servicio se está ejecutando y por lo tanto se ha creado un objeto que la aplicación devolverá a la salida.
5. Una vez finaliza la ejecución de la aplicación, el servidor devuelve los datos generados al cliente y una serie de datos de información donde se indica el tipo de datos que el cliente va a recibir.

Una aplicación de estas características se puede implementar en diversos lenguajes de programación, siempre y cuando se pueda generar un fichero ejecutable. Los más habituales son C, C++, Java o Visual Basic entre otros. Pero es más sencillo si se realiza con un lenguaje que ofrezca una implementación sencilla con cadenas de texto, un lenguaje que cumple esta especificación es Perl que aporta operaciones sencillas de gestión de cadenas de texto.

Algunos tipos de CGIs pueden ser:

- Contador de acceso: Número de solicitudes que una página ha tenido.
- Buscador: Encuentra las páginas que tienen unas palabras determinadas.
- Correo: Permite recuperar datos de información del usuario.
- Contribuciones: Introducir notas de información o enlaces en una página.
- Estadísticas de uso: Datos informáticos de los eventos ocurridos en el servidor.
- Administración remota del servidor: Ofrece la posibilidad de ejecutar los programas que gestionan el procedimiento de actuación del servidor.

2.5.2. JAVA SERVLET

Un servlet[10] es un objeto de tipo Java que son ejecutados tanto fuera como dentro de un contenedor de servlets, que es un programa que recibe las peticiones del cliente y las redirecciona al objeto de tipo servlet. Su función es procesar las peticiones del cliente y crear las determinadas respuestas dependiendo de la petición que se ha efectuado.

Comúnmente, son utilizados para crear páginas web de forma dinámica a través de los datos que el cliente envía al servidor mediante la petición.

El esquema de comunicación que sigue un intercambio de información de este tipo es el siguiente:

1. Se lee la petición del cliente con los datos que ha enviado para procesar la petición.
2. A continuación se obtienen los datos correspondientes a la petición que están encapsuladas dentro de ella.
3. Una vez se conoce toda la información necesaria, se procede a obtener los datos requeridos por el cliente, por lo que es posible que se realice alguna conexión con una base de datos.
4. Posteriormente se genera la respuesta que va a ser devuelta al cliente siguiendo un formato estándar de comunicación.
5. Finalmente se envía al cliente el mensaje de respuesta con la información que había sido requerida a través de la petición.

Para realizar las peticiones mediante este tipo de implementación se dispone de dos tipos:

- GET: Este tipo de petición manda los datos en la propia URL (Uniform Resource Locator [24]), por lo que son visibles para el usuario en un navegador. Esto hace que este tipo de petición sea muy vulnerable a intrusiones. En caso de hacer gestiones con una base de datos, este tipo de obtención de información supone un riesgo de estabilidad de la base de datos, pudiendo generarse o modificarse registros simplemente enviando peticiones a través de la barra de direcciones de un navegador web.

- POST: Este modelo de conexión manda los datos en la cabecera de la petición de manera que quedan invisibles al usuario. Es por ello que las peticiones que requieren un cierto grado de seguridad se manden utilizando este parámetro.

En el servidor estos dos parámetros se gestionan a través de dos métodos diferentes, dado que se obtiene la información que manda el cliente de diferentes zonas de la petición.

2.6 BASE DE DATOS

Una base de datos[11] es un conjunto de datos que están almacenados siguiendo una estructura establecida de un mismo contexto. Por lo tanto, todo proyecto que realice operaciones con un gran conjunto de datos necesitará esta herramienta para la gestión de esa información.

Hay diversos tipos de bases de datos según su modelo de administración de datos:

- Jerárquicas: Organización en forma de árbol invertido. Utilizada para aplicaciones con gran cuantía de datos. Incapaz de representar la redundancia de información.
- De red: Es un modelo variante del jerárquico en el que un nodo (hijo) del árbol puede tener varios padres. Soluciona el problema con la redundancia de datos
- Transaccionales: Uso poco común. Tiene como finalidad una gran velocidad de gestión de datos, tanto envío como recepción.
- Relacionales: Se basa en relacionar las tablas de la base de datos. Modelo mas utilizado para gestionar los datos de forma dinámica. Los datos se almacenan u obtienen mediante consultas, que se realizan, de forma más habitual, en lenguaje SQL (Structured Query Language [22] o Lenguaje Estructurado de Consultas).
- Multidimensionales: Son similares a las relacionales pero se diferencian a nivel de concepto. Tienen una única tabla en la que hay un campo o columna por cada dimensión, y por cada métrica existe otro campo.
- Orientadas a Objetos: Está basada en el modelo de programación orientado a objetos, de forma que en la base de datos se guardan todos los elementos de un objeto. Contiene propiedades como la encapsulación (ocultación de información), herencia (heredar comportamientos), polimorfismo (aplicación de una operación a diferentes objetos).
- Documentales: Permiten guardar la información de forma completa, como un documento. Este tipo de datos puede ser XML, JSON o documentos con formato Microsoft Office o PDF (Portable Document Format [16]).
- Deductivas: Basada en el almacenamiento de hechos o reglas. Conocidas también como bases de datos lógicas debido a que la gestión se realiza siguiendo lógica matemática.

De todos estos modelos, se va a realizar un estudio más detallado de dos de ellos, los cuales se han considerado más adecuados para la gestión de la información debido a la estructura de la aplicación.

2.6.1. BASES DE DATOS RELACIONALES

Como se ha dicho anteriormente, estas bases de datos siguen el modelo relacional[12] de manera que las tablas están conectadas entre si, es decir están relacionadas a través de alguno de sus campos. Están compuestas de tablas y relaciones que no pueden estar duplicadas, es decir no pueden tener el mismo nombre. Estas tablas contienen registros compuestos de filas y de columnas.

Para relacionar varias tablas se realiza a través de registros determinados denominados clave primaria (registro en la tabla padre) y foránea (registro de la tabla hija).

Existen restricciones que limitan el tipo de datos de los registros, es decir, a cada columna de un registro se le puede imponer una limitación de forma que sea un campo numérico donde solo se puedan guardar dígitos. También se puede limitar la longitud de la cadena que se va a guardar de forma que no se puedan guardar datos de mayor longitud.

Las claves únicas que tiene una tabla relacional sirven para identificar registros de una misma tabla de forma que sean identificados a través de un campo único que no puede tomar dos valores idénticos.

De todas las claves únicas de las que puede disponer una tabla, se escoge una que pasa a llamarse clave primaria y es la que define a todos los demás campos de la tabla, de manera que es el campo que se relaciona con las otras tablas de la misma base de datos.

Para completar la relación de dos tablas se hace mediante una clave foránea, que es la referencia a la clave primaria en otra tabla, es decir, es un campo de una tabla que va a ser referenciada desde la tabla padre.

En la *Ilustración 7* se puede ver una relación de tablas mediante el campo 'ID_Usuario'.

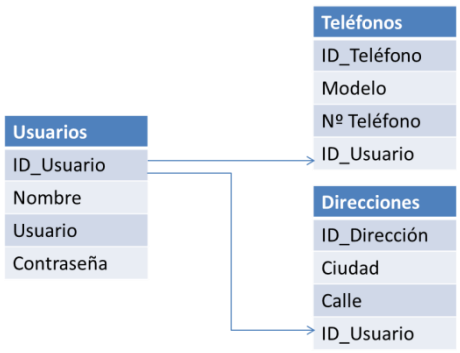


Ilustración 7. Relación de BD relacional

Las bases de datos relacionales permite aumentar la velocidad de acceso a los datos mediante claves índice, de modo que se recuperen los datos de forma más rápida al hacer filtros de búsqueda por este índice.

La estructura de una base de datos se divide en dos partes:

- Esquema: Se compone de los siguientes elementos:
  - Nombre de las tabla
  - Nombre de cada campo (columna)
  - Tipo de datos de cada campo
  - Tabla a la que pertenece cada campo
- Datos: Son cada campo de un registro de información de la base de datos, es decir cada valor de un campo de una tabla es un dato.
- Para obtener los registros de información de una base de datos relacional se utiliza el algebra relacional y el cálculo relacional de forma que mediante el primero se formula una consulta y a través del segundo se establece lo que se quiere recuperar. Normalmente se utiliza lenguaje SQL para realizar las consultas.

2.6.1.1. MySQL

MySQL[13] es un sistema de gestión de bases de datos que siguen el modelo relacional. Es un software libre que es compatible con cualquier implementación que use la licencia GNU GPL (GNJU is Not UNIX General Public License [8]). A pesar de ser software libre, si se va a utilizar para desarrollar productos privativos, se requiere la compra de una licencia para su utilización. El código es patrocinado por una empresa privada que tiene la gran mayoría del código con copyright.

Para acceder a una base de datos MySQL se puede desarrollar aplicaciones en multitud de lenguajes de programación como pueden ser C, C++, C#, Java, Pascal, PHP etc ya que son compatibles con este tipo de gestión de base de datos.

Aporta unas medidas de seguridad importantes debido a que se requiere una identificación para la conexión a cualquier base de datos, aportando una seguridad en la conectividad.

Tolera un gran conjunto de datos llegando hasta los 50 millones de registros.

Tiene una gran variedad del lenguaje SQL de forma que algunas extensiones también son incluidas.

Es compatible con una gran variedad de sistemas.

Ofrece un servicio de replicación que consiste en la realización de una copia de seguridad da los datos para subsanar posibles fallos, por lo que siempre deberá estar actualizada.

2.6.2. BASES DE DATOS DOCUMENTALES

Una base de datos documental[14] está basada en un conjunto de programas capaces de gestionar datos de forma estructurada como documentos. Son bases de datos No SQL y son una de las principales dentro de este grupo de bases de datos.

Están orientadas a almacenamiento de documentos, de forma que, dado un documento, se encapsula la información en un tipo de datos y se codifica a través de un formato estándar para su almacenamiento. Algunos de estos estándares son XML o JSON.

La estructura a la hora de guardar los datos no es tan estricta como en una base de datos relacional de forma que en una base de datos orientada a documentos no tienen que seguir un mismo esquema ni tener todos los registros los mismos atributos o claves.

Para recuperar los documentos se hacen mediante claves que los identifican, normalmente es una cadena de texto que está incluida en un índice en la base de datos, de forma que la recuperación de registros sea rápida.

Además de esta clave que identifica a los documentos existe una API proporcionada por la base de datos que permite recuperar la información mediante el contenido de los registros, es decir, poder buscar en la base de datos a través del contenido de los documentos.

2.6.1.2. MongoDB

MongoDB[15] es un sistema de base de datos no estructura No SQL, orientado a documentos, es decir, es un tipo de base de datos documental. Al igual que MySQL es software libre, es decir de código abierto, disponible para cualquier programador que quiera hacer uso con libertad utilizando la licencia publica de AGPL (Affero General Public License $_1$).

Este tipo de base de datos no utiliza tablas con relaciones, como pasa con las bases de datos SQL, si no que guarda los documentos de manera estructurada siguiendo un esquema con formato JSON, que en este lenguaje es conocido como BSON o Binary JSON.

Ofrece la posibilidad de realizar búsquedas mediante campos, rangos o expresiones regulares de forma que la respuesta de la consulta puede ser tanto un campo determinado del documento como una función creada por el programador.

Permite añadir índices a los campos de los documentos registrados, facilitando la búsqueda en los diferentes registros.

Al igual que MySQL, MongoDB ofrece un servicio de replicación de forma que se realiza una copia de seguridad en un servicio esclavo de servicio principal, pero éste no tiene

permisos para realizar modificaciones en la base de datos maestro. Para ello se puede realizar un balanceo de carga de manera que se distribuyan los datos en una colección de forma que se pueda garantizar una seguridad en caso de fallo del sistema.

Es compatible con multitud de lenguajes como C, C++, C#, JavaScript, Java, PHP para la implementación de las consultas.

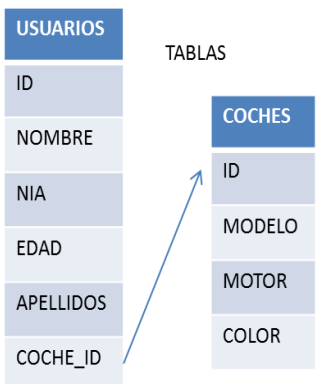
Para el almacenamiento de los registros en la base de datos no se sigue una estructura determinada ya que los datos son documentos que se guardan formando colecciones. Estos documentos son conjuntos de datos que pueden tener cada uno un esquema distinto y que puede verse modificado en cualquier momento tanto el valor de los datos como los nombres de los capos. Están compuestos de elementos denominados clave-valor de forma que se la clave es el nombre que identifica el campo y el valor es el dato guardado que puede ser cualquier carácter.

Un ejemplo de almacenamiento de un documento en MongoDB es el mostrado en la *Tabla 1*.

Registro 1	Registro 2
{ "ID":"1", "Nombre":"David", "NIA":"100066661", "Edad":"24", "Coche": { "Modelo": "SEAT Leon", "Motor": "1.9" "Color": "Blanco" } }	{ "ID":"2", "Nombre":"Iria Manuela", "Apellidos":"Estevez Ayres" }

Tabla 1. Almacenamiento BD documental

Mientras que un almacenamiento de los mismos datos en MySQL es el mostrado en la *Tabla 2* donde se visualiza el esquema de relación de la base de datos y las tablas con los registros.



COCHES

#	ID	MODELO	MOTOR	COLOR
1	1	SEAT Leon	1.9	Blanco

USUARIOS

#	ID	NOMBRE	APELLIDOS	NIA	EDAD	COCHE_ID
1	1	David	NULL	10006661	24	1
2	2	Iria	Estevez Ayres	NULL	NULL	NULL

Tabla 2. Almacenamiento BD documental

Tanto MongoDB, como MySQL, ofrecen aplicaciones con interfaces gráficas para facilitar la gestión de las bases de datos, además de ciertos comandos que se pueden introducir desde una consola para la gestión de la bases de datos.

2.7 CONCLUSIONES

Por lo tanto, para finalizar, se establece como requisito, por parte de la tutora, el desarrollo de la interfaz en el sistema operativo Android.

La transmisión de datos entre el cliente y el servidor se realiza utilizando el formato JSON para la codificación de los datos, debido a que es más sencillo y resulta más fácil hacer un analizador sintáctico para codificar o decodificar este lenguaje.

Se decide el uso de un java servlet debido a que consume menos recursos que CGI y es más eficiente al no realizar un proceso diferente cada vez que se realiza una petición, si no que hay un registro de los datos en la máquina y cada vez que el usuario envía una solicitud, se consulta ese registro lo que supone un consumo menor de memoria y una ejecución más rápida. Los programas en CGI, una vez finalizada la ejecución, se cierran, de forma que cada vez que se realiza una petición el programa se inicia de nuevo, mientras que utilizando un servlet se ahorra memoria al estar los programas cargados en la memoria. Para ejecutar el java servlet se usa el contenedor de servlets conocido como Apache Tomcat.

Para la organización de los dos sistemas de gestión de base de datos se decide usar una base de datos relacional, dada la estructura que sigue el proyecto y la facilidad de gestión de datos a la hora de relacionar los usuarios con los grupos y éstos a su vez con los datos de cada módulo, correspondiente a cada grupo de la aplicación.

CAPÍTULO 3. DISEÑO

3.1 INTRODUCCIÓN

La finalidad del proyecto es el desarrollo de una aplicación que gestione los grupos de prácticas de los alumnos de la Universidad Carlos III de Madrid.

Para ello se implementaran dos aplicaciones de comunicación, que facilitarán a los alumnos la organización de las prácticas a lo largo del curso.

Para la implementación de este proyecto se necesita diseñar por lo tanto una aplicación para un Smartphone, un servidor, una base de datos y un sistema de conexión entre cada módulo como se muestra en la *Ilustración 8*.

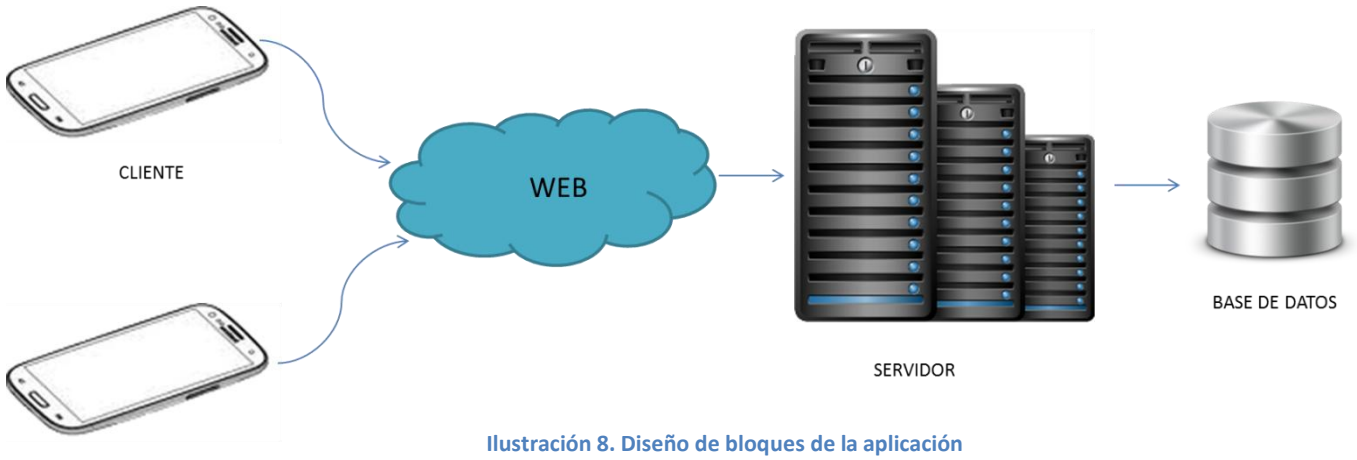


Ilustración 8. Diseño de bloques de la aplicación

3.2 REQUISITOS

A continuación se describen los requisitos que debe de cumplir la aplicación para las diferentes secciones de ésta.

1. Proceso de identificación
    a. Tiene que haber un registro de usuarios en la base de datos
    b. Se debe de introducir obligatoriamente el NIA y la contraseña
    c. El alumno debe introducir el usuario y la contraseña correctamente

2. Grupos registrados
    a. Se muestra una lista de grupos en los que el alumno, que se ha identificado previamente, está registrado, en caso de existir alguno.
    b. Se da la opción de crear un grupo nuevo
    c. Se notifica la existencia de grupos pendientes así como la posibilidad de acceder a la sección que los muestra.
3. Creación de nuevo grupo
    a. Es necesario introducir obligatoriamente el nombre del grupo
    b. Por defecto, se añade al creador como nuevo miembro del grupo.
    c. Se ofrece la posibilidad de añadir otros miembros al grupo
    d. Esos miembros no podrán estar ya introducidos
    e. Debe de existir el alumno en la base de datos
    f. El campo del miembro no podrá estar vacío a la hora de añadir uno nuevo.
    g. Al crear el grupo se deben realizar las peticiones pertinentes y debe de aparecer el grupo en los registrados del creador.
4. Grupos pendientes
    a. Se muestra la lista de grupos que están pendientes de aceptar por un usuario
    b. Se da la opción de aceptar un grupo y por lo tanto pasar a la lista de registrados.
5. Grupo
    a. Se mostrarán las diferentes secciones de un grupo
6. Chat
    a. Se deben cargar los mensajes guardados en la memoria interna del teléfono.
    b. Se hace una carga de los mensajes que han mandado otros usuarios y están pendientes de recepción.
    c. El mensaje a enviar no debe de estar vacío.
    d. Se muestra el usuario que envía el mensaje y la hora a la que se envía para llevar una mejor gestión de los mensajes en el tiempo.
    e. Estando el chat en primer plano se realiza una carga periódicamente de los mensajes que han sido enviados por otros usuarios.
7. Agenda
    a. Se muestra un calendario, en un modulo, al usuario con el día actual seleccionado
    b. Al seleccionar un día se muestran los eventos programados para ese día.
    c. En otro modulo se listan los eventos que tiene asignados ese grupo.
    d. Al seleccionar un evento en el segundo módulo se puede visualizar los detalles del evento.

e. En este módulo se permite crear un nuevo evento introduciendo obligatoriamente el asunto, la localización y la fecha, siendo la hora opcional.
f. No se permite crear citas con asuntos duplicados.
g. Los eventos que han ocurrido ya, se borraran automáticamente de la base de datos, realizando una comprobación cada día.

8. Información del grupo
    a. Se muestra el nombre del grupo y los miembros registrados en él.
    b. Se ofrece la posibilidad de añadir nuevos al grupo con las mismas restricciones que en el caso de crear un nuevo grupo, es decir, que el campo no esté vacío, que el usuario exista en la base de datos y que no esté ya introducido.
    c. Se puede abandonar un grupo borrando así todos los registros correspondientes a ese usuario para ese grupo en la base de datos, y borrando los datos guardados en la memoria interna del teléfono. Si es el único miembro que queda en el grupo se borra todo lo relacionado con ese grupo en la base de datos, tanto mensajes de chat como eventos del calendario.
9. Submenú
    a. Se muestra un submenú en todas las ventanas, excepto en la de identificación inicial, donde el usuario podrá recargar la información de la pantalla actual, volver al listado de grupos registrados, leer una ayuda de la aplicación y ver la información de los desarrolladores del proyecto.

3.3 DISEÑO DEL CLIENTE

El primer paso que se da a la hora de realizar un diseño de una interfaz es hacer una lista de las pantallas que se van a querer desarrollar y con las funcionalidades que se deseen implementar. A continuación se realiza un esbozo sobre papel de un posible diseño de cada una de las pantallas que la aplicación va a tener y cuál es el ciclo de vida de cada una.

En la *Ilustración 9* se muestra el principal diseño de las pantallas, donde se puede observar que la aplicación se divide en tres bloques principales: la identificación, la gestión de grupos y la organización de un grupo concreto.

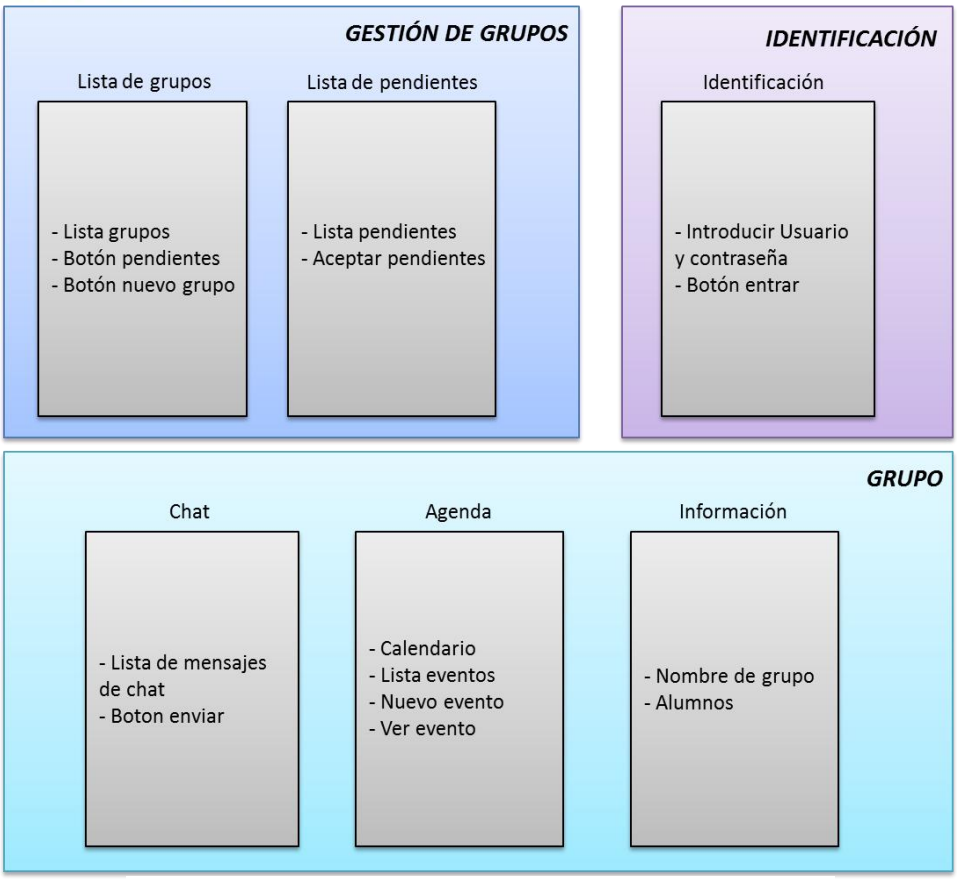


Ilustración 9. Diseño principal de interfaz

Una vez se tiene el diseño del esquema de las pantallas de la aplicación se procede a su implementación de forma que se obtiene una aplicación en la que se puede realizar una navegación entre las diferentes ventanas pero sin ninguna funcionalidad de guardado persistente de datos implementada aun.

3.4 DISEÑO DEL SERVIDOR

Este modulo es el encargado de procesar las peticiones que realiza el cliente para posteriormente devolverle los datos requeridos.

Se divide la implementación de este bloque en la gestión de tres secciones claramente diferenciadas como se puede observar en la *Ilustración 10*

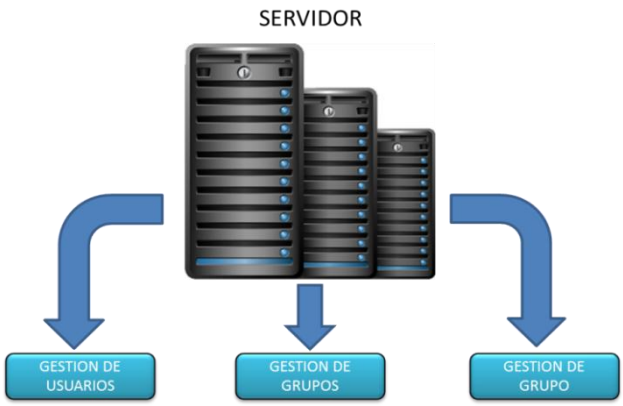


Ilustración 10. Bloques del servidor

- Gestión de usuarios. Este bloque se encarga de la gestión de operaciones con los usuarios como el inicio de sesión o la comprobación de existencia de un usuario.
- Gestión de grupos. En este modulo se organizan todas las instrucciones referentes a los grupos, como la creación de grupo.
- Gestión de grupo. Sección de operaciones relacionadas con un grupo concreto, como los mensajes de chat o los eventos.

3.5 DISEÑO DE LAS CONEXIONES CLIENTE-SERVIDOR MODULOS

Para la tramitación de las diferentes peticiones que el cliente pueda realizar se establecen una serie de comandos que identifiquen cada operación. En la *Ilustración 11* se muestra un esquema de la conexión entre ambos bloques.

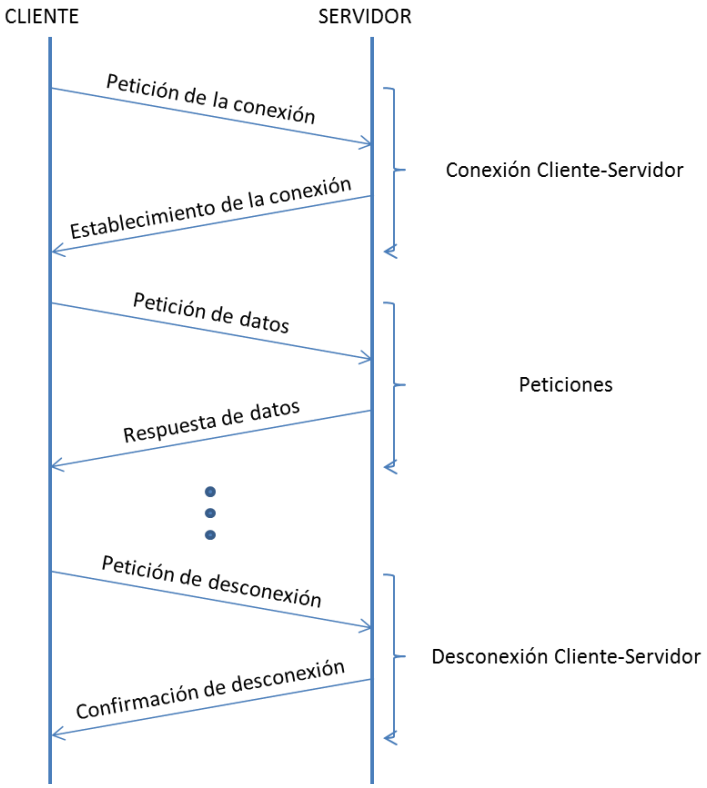


Ilustración 11. Diagrama de comunicación Cliente - Servidor

Apuntar que en cada petición que se efectúa, se realiza otra petición desde el servidor a la base de datos para obtener los datos, de forma que en la imagen solo se muestra el primer paso de la conexión de la aplicación.

3.6 DISEÑO DE LA BASE DE DATOS

Primeramente se realiza un estudio de los registros que se van a almacenar en la base de datos.

Para ello se hace una lista con los campos que se van a guardar y posteriormente se agrupan siguiendo una disposición de tablas, que se organizan como una estructura ordenada.

Esta organización se muestra en la *Ilustración 12*, donde se ve el resultado de colocar los datos según se van a realizar las consultas dependiendo de las secciones de la aplicación.

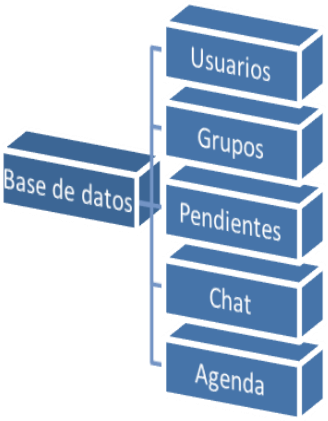


Ilustración 12. Organización BD

- Gestión de usuario. Para organizar estos datos se utiliza una única tabla de usuarios que contendrá los datos de identificación.
- Gestión de grupos. Las tablas de 'grupos' y 'pendientes' son las utilizadas para almacenar los datos necesarios para realizar operaciones con grupos.
- Gestión de un grupo determinado. La organización de un determinado grupo se hace a través de las tablas 'chat' y 'agenda' donde se gestionaran los datos correspondientes a los distintos módulos de cada grupo.

A continuación se procede a diseñar la estructura interna de cada tabla, es decir, establecer el nombre de las columnas de las tablas y el tipo de datos de cada una de las diferentes tablas.

3.7 DISEÑO DE LAS CONEXIONES SERVIDOR-BASE DE DATOS

El servidor realiza una serie de consultas a la base de datos para obtener los registros que el cliente solicita. En la *Ilustración 13* se muestra un diagrama que muestra el flujo de peticiones y respuestas entre el servidor y la base de datos.

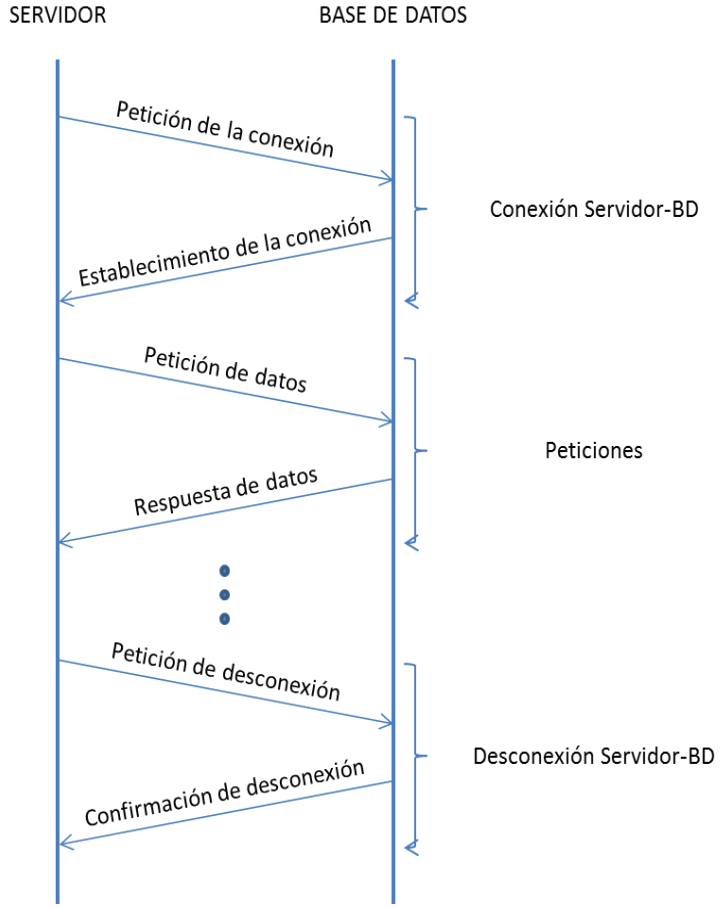


Ilustración 13. Diagrama de comunicación Servido – Base de Datos

Cada una de las peticiones que el servidor realiza a la base de datos se realizan utilizando sentencias básicas del lenguaje MySQL tales como 'INSERT' para crear nuevos registros, 'DELETE' para borrar algún registro o 'SELECT' para obtener los datos.

3.8 CONCLUSIONES

Para concluir, una vez se ha realizado el diseño, se puede observar que el proyecto está dividido en tres bloques principales completamente diferenciados, que necesitan de una conexión entre ellos para el completo funcionamiento de la aplicación. Para cada uno de ellos, se ha establecido un primer diseño, en el que se esquematizan las funciones que se han determinado en el inicio del desarrollo de este proyecto, ya que posteriormente van a surgir nuevas implementaciones que no se han tenido en cuenta al principio.

CAPÍTULO 4. IMPLEMENTACIÓN

En este capítulo se especifican los medios y los programas utilizados para el desarrollo de la aplicación, así como una explicación detallada de la implementación de cada bloque.

4.1 MEDIOS Y PROGRAMAS

En esta sección se explican los medios y los programas que han sido necesarios a lo largo de la realización del proyecto.

4.1.1 MEDIOS

Los medios utilizados para la implementación de las diferentes secciones del proyecto son los siguientes:

- Ordenador de sobremesa:
  - Placa base Asus P8H61-M LE/USB3
    - Procesador Intel Core i7 -2600 3.4GHz (Giga Hertz [6])
    - Memoria 8 GB (GigaByte [4]) RAM (Random Access Memory [17])
  - Sistema Operativo Windows 7
  - Tarjeta gráfica NVIDIA GeForce GTX560
  - Disco duro de 2 TB (TeraByte [23])
  - 2 Monitores 22”
- Ordenador portátil Acer Aspire 7540G:
  - Procesador AMD Athlon II Dual-Core M300 2 GHz
  - Memoria 4 GB RAM DDR2 a 667 MHz (Mega Hertz [15])
  - Sistema Operativo Windows 8
  - Tarjeta gráfica ATI Mobility Radeon HD 4570 512 Mb DDR3-SDRAM
  - Disco duro de 640 GB
  - Pantalla de 17.3”

4.1.2 PROGRAMAS

Los programas utilizados para el desarrollo del proyecto se dividen en dos bloques y son los siguientes:

- Implementación de interfaz Android:
  - Eclipse: Programa empleado para el desarrollo del código fuente de la aplicación del cliente.
  - Microsoft Picture It!: Programa de retoque fotográfico utilizado para el diseño de las imágenes utilizadas en la interfaz del cliente.
- Implementación servidor y base de datos:
  - VirtualBox: Máquina virtual en la que se simula el sistema operativo de Linux donde se ejecuta el servidor y donde se encuentra la base de datos. Tiene las siguientes características:
    - Versión 2.4.8
    - SO simulado: Ubuntu 12.04
    - Memoria base: 1024
    - Memoria de video 64 MB (MegaByte $_{14}$)
    - Espacio de disco dedicado: 10 GB

    Dentro de la máquina virtual se hace uso de las siguientes herramientas:

    - Workbench: Programa para la gestión de la base de datos a través de una interfaz gráfica.
    - Eclipse: Programa de desarrollo en el que se implementa el código del servidor.
    - Apache Tomcat: Es un servidor de código abierto sobre el que se ejecutará el código implementado
    - MySQL: Sistema de gestión de bases de datos relacional.
    - JDK 7: Software que proporciona las herramientas necesarias para el desarrollo de aplicaciones en lenguaje JAVA.
- Desarrollo de la memoria:
  - Microsoft Office: Paquete de aplicaciones de la empresa Microsoft que contiene herramientas como editores de texto, tablas, gráfica, presentaciones, etc.
  - Microsoft Project: Programa de administración avanzada de proyectos.

4.2 FASES DE DESARROLLO

El desarrollo de la aplicación se divide en cuatro fases principales: estudio de posibilidades y viabilidad del proyecto, diseño e implementación de la interfaz, diseño e implementación del servidor y la base de datos y la fase de pruebas y mejoras

4.2.1 ESTUDIO DE POSIBILIDADES Y VIABILIDAD DEL PROYECTO

El primer paso para la realización del proyecto es presentar una serie de ideas de posibles proyectos a desarrollar.

Una vez se tiene una lista de posibilidades, se debe realizar un estudio de las ideas y escoger las que sean más útiles y originales de la lista, sin descartar las demás ideas ya que pueden ser utilizadas en mejoras futuras de la aplicación.

Para finalizar el estudio previo del proyecto se realiza una investigación de las posibles tecnologías que se pueden utilizar en un proyecto de esas características y se seleccionan las mejores para un desarrollo óptimo del proyecto.

4.2.2 DISEÑO E IMPLEMENTACIÓN DE LA INTERFAZ

Una vez se han escogido las mini-aplicaciones que se van a desarrollar para el proyecto se procede a realizar el diseño de la interfaz del cliente.

Primeramente se hace un esbozo de cómo serán las pantallas y el diagrama de estados de cada una. En la *Ilustración 14* se muestra el diagrama de flujo, donde cada estado es una ventana donde se realizaran las diferentes operaciones.

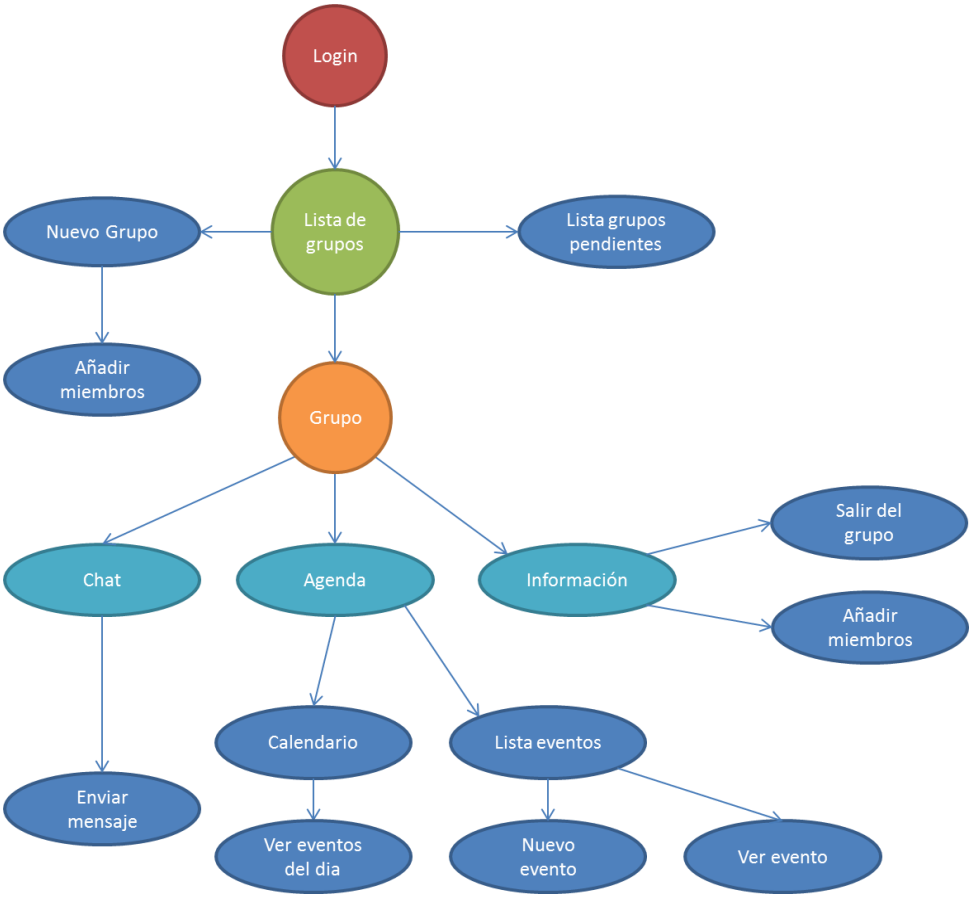


Ilustración 14. Diagrama de estados

Una vez está claro el diseño se procede a la implementación de la interfaz. El desarrollo se realiza por partes debido a la complejidad de algunas pantallas. A continuación se detalla el proceso de los diferentes proyectos:

- Desarrollo de la pantalla de identificación y pantalla principal de lista de grupos.
- Implementación de la ventana de grupos con la barra inferior de tres subsecciones básicas y el submenú de la aplicación.
- Implementación de las mini-aplicaciones que contiene el proyecto.
- Unión de todas las partes.

Las razones principales de esta división de tareas son la dificultad de realizar la interfaz de la mini-aplicación del calendario y hacer una pantalla común para las diferentes mini-aplicaciones que serán seleccionadas desde una barra situada en el inferior de la ventana.

Cuando ha finalizado la implementación de toda la interfaz, se realiza una serie de pruebas de funcionamiento básico de las pantallas de la aplicación.

4.2.3 DISEÑO E IMPLEMENTACIÓN DEL SERVIDOR Y LA BASE DE DATOS

Acabado el desarrollo de la interfaz del cliente a falta de la conexión con el servidor se procede a diseñar el servidor y la base de datos.

En este punto se conocen las variables que el cliente envía al servidor, tanto para consultas como para creaciones o borrados de registros. Se diseñan todos los métodos necesarios para las peticiones que realiza el cliente y se definen los parámetros que el cliente envía para realizar las consultas.

Una vez se han diseñado los métodos requeridos en el servidor y se tienen claros los campos necesarios para la aplicación se procede a diseñar la base de datos.

Esta fase no es demasiado complicada ya que si el estudio anterior y el diseño se han desarrollado correctamente, solo es agrupar todas las variables, es decir, se agrupan todas las variables que se quieren guardar y todas las variables que se van a leer de cada petición que realiza el cliente. Posteriormente se agrupan en bloques que estén relacionados, de tal forma que se organicen según la aplicación a utilizar.

Diseñados ambos bloques se procede a implementar cada uno por separado.

Primeramente se realiza el desarrollo del servidor donde se implementa un método que procesa la petición del cliente y diferentes procesos que gestionan las diferentes instrucciones del cliente. Cada proceso supone como mínimo, una conexión con la base de datos para obtener o insertar registros.

Habiendo finalizado el servidor, se continúa con la creación de la base de datos. Se puede realizar haciendo uso de la consola del sistema o mediante algún programa de

gestión de bases de datos para hacerlo mas sencillo. Una vez se han creado las diferentes tablas, se crean registros a mano para las pruebas posteriores.

Concluidas las implementaciones de los dos bloques por separado se realiza la conexión del servidor con el cliente añadiendo las peticiones necesarias en el código fuente del cliente de manera que queden todos los bloques unidos.

A continuación se realizan una serie de pruebas de conexión con el servidor para comprobar que el enlace se realiza correctamente.

Verificada esta conexión se procede a realizar la conexión del servidor con la base de datos. Para ello se añaden las diferentes consultas a la base de datos en cada método del servidor.

Después de tener todos los bloques unidos se ejecuta una serie de pruebas básicas de todo el conjunto del proyecto, comprobando que todos los enlaces son correctos, haciendo más hincapié en la última unión de la base de datos, verificando que se recuperan e insertan bien los parámetros.

De esta manera se deja finiquitada una primera versión de la aplicación.

4.2.4 FASE DE PRUEBAS Y MEJORAS

Acabada la unión de todos los bloques se procede a hacer una revisión exhaustiva de todo el proyecto. Esta serie de pruebas se realiza desde diferentes dispositivos para comprobar el correcto funcionamiento de la aplicación. Se dispone de varios usuarios que realizarán estas pruebas para optimizar la búsqueda de posibles fallos en la aplicación.

Una vez se han realizado las pruebas los 'probadores' dan un informe a los desarrolladores en donde notifican los errores encontrados y las posibles mejoras que se podrían introducir en la aplicación.

A continuación se procede a corregir los fallos encontrados y a implementar una serie de mejoras de las propuestas por los 'probadores'.

Acabado el proceso de arreglo del código se vuelve a enviar la aplicación a los 'probadores' para que, de nuevo, se hagan pruebas de la aplicación completa. Se hace primeramente una revisión de los fallos encontrados en la primera ronda de pruebas y posteriormente se hace una regresión de la aplicación para comprobar que no se han generado fallos nuevos al implementar el nuevo código para arreglar los errores fallos encontrados.

4.3 ESQUEMAS Y FUNCIONAMIENTO

Como se ha comentado en el capítulo anterior, la aplicación consta de tres bloques principales y la comunicación entre ellos. A continuación se detalla cada sección:

4.3.1 ESTRUCTURA DEL CLIENTE

4.3.1.1 Identificación

En primer lugar el usuario debe de iniciar sesión para poder utilizar la aplicación

Como requisito se establece necesario rellenar el campo de usuario y de contraseña para su posterior consulta de registro en la base de datos, en caso contrario se mostrará un cuadro de diálogo en el que se informa de la necesidad de cumplimentar esos campos.

Se muestra un error tanto en caso de no estar registrado, ya que solo podrán utilizar esta aplicación los alumnos de la Universidad Carlos III de Madrid, como en el caso de no haber introducido correctamente los datos de inicio de sesión.

Una vez el proceso de identificación se ha desarrollado con éxito, se guarda el usuario como una constante que será fija a lo largo de toda la ejecución, ya que se utilizará para diversas consultas en la base de datos.

4.3.1.2 Gestión de grupos

Una vez iniciada la sesión, se abre la pantalla principal de la aplicación que muestra la lista de los grupos a los que está subscrito el usuario, la existencia de invitaciones a grupos y la posibilidad de crear un nuevo grupo.

4.3.1.2.1. Lista de Grupos

El usuario visualiza una lista con los nombres de los grupos en los que está registrado.

Al realizar una pulsación sobre cualquier nombre de grupo, accede a éste para poder ejecutar cualquiera de las posibles funcionalidades implementadas sobre un grupo.

4.3.1.2.2. Nuevo grupo

El usuario puede crear un nuevo grupo en el que al menos estará él como integrante. Se le pide que introduzca obligatoriamente un nombre para el grupo y opcionalmente puede añadir más miembros al grupo de manera individual a través de una ventana emergente.

Hay una serie de restricciones cuando se intenta crear un grupo nuevo:

- No se permite dejar ningún campo vacío. En caso del nombre del grupo no se permite que sea una cadena vacía, es decir, un nombre en blanco. Si el nombre del grupo está vacío se le

notifica al usuario por pantalla el error correspondiente. En el caso de la lista de los alumnos, no surge este problema, porque por defecto se añade al propio usuario de forma que no está vacía nunca, pero a la hora de querer añadir un nuevo miembro también se comprueba que el campo no esté vacío.

- Alumno inexistente. A la hora de añadir un nuevo miembro al grupo, se comprueba consultando al servidor, que existe el usuario en la base de datos, para así evitar que se creen registros que no van a ser leídos nunca. En caso de no existir, se le comunica al usuario mediante un mensaje.
- Alumno repetido. Después de comprobar que el alumno existe en la base de datos, se comprueba que ese miembro no ha sido añadido ya a la lista. Si ya se encuentra en ella se le avisa al usuario.

Una vez se ha creado el grupo después de pulsar el botón habilitado para ello, se vuelve a mostrar la ventana principal.

4.3.1.2.3. Grupos pendientes

Se muestra en un botón la disponibilidad de peticiones a nuevos grupos que tiene el usuario. Estas invitaciones son las generadas a la hora de crear un grupo por parte de otro alumno. Este botón muestra el estado de las peticiones, es decir, en el caso de haber nuevas, el botón tiene el texto 'Hay grupos pendientes', mientras que en caso de que no haya invitaciones, muestra 'No hay grupos pendientes'.

Si el usuario pulsa el botón, se le muestra una nueva pantalla en la que se listan las nuevas peticiones.

Lista de pendientes

Se observa una lista de invitaciones a diferentes grupos, donde el usuario ve únicamente el nombre del grupo, no conoce los integrantes ni quien le ha invitado, es decir, no dispondrá de ninguna información del grupo mientras no acepte la petición.

Una vez se selecciona un grupo pendiente se le plantea, a través de un cuadro de dialogo, la opción de aceptar dicha invitación o cancelar la acción, pero no rechazar la invitación en ningún momento.

4.3.1.3 Grupo

Hay que destacar que los grupos se gestionan a partir de un identificador para que no existan problemas a la hora de crear grupos con el mismo nombre y a la hora de obtener la información de los grupos. Este identificador se asigna automáticamente en el servidor y es único para cada grupo. Este proceso de

asignación y de gestión por identificadores es totalmente invisible al usuario, ya que ni siquiera conoce su existencia.

En esta pantalla el usuario podrá ver las diferentes opciones del grupo a través de los diferentes módulos de la barra inferior de la pantalla. La estructura que sigue un grupo es la mostrada en la *Ilustración 15*

Ilustración 15. Esquema de un grupo

4.3.1.3.1. Chat

Esta sección es la primera que se muestra cuando un grupo es seleccionado desde la lista.

Esta funcionalidad permitirá al usuario comunicarse con los demás participantes del grupo de manera que los usuarios recibirán los mensajes cada vez que entren en un grupo y se posicionen en la pestaña de chat. En caso de que varios usuarios del mismo grupo estén dentro del chat al mismo tiempo, se considera una mensajería instantánea, ya que la obtención de los mensajes se está realizando constantemente si la pantalla del chat está en primer plano.

El mensaje que se manda al grupo no puede estar vacío, al menos tiene que contener un carácter para poder ser enviado, en caso contrario, se le mostrará un mensaje al usuario.

Se les muestra a los usuarios la hora a la que el mensaje fue enviado para tener un mayor control de la conversación.

Cada vez que un usuario manda un mensaje, se guarda un registro en la base de datos para cada miembro del grupo, incluido para el que lo manda, de manera que los demás usuarios lo recibirán en el momento en el que

entren a la pantalla del chat y el propio usuario que lo manda lo recibirá instantáneamente.

El almacenamiento de los mensajes del chat se realiza en el propio dispositivo a través de un fichero de texto para cada grupo, que será borrado cuando se desee salir del grupo o cuando se desinstale la aplicación.

Mediante esta gestión de la mensajería se ahorra en la capacidad de registros y tiempo de consultas de la base de datos ya que únicamente se guardan los mensajes que están pendientes de envío y no la totalidad de los mensajes de la conversación.

A la hora de mostrar los mensajes, el cliente lee el fichero de texto y carga los diez últimos mensajes que se enviaron, posteriormente realiza una llamada al servidor para obtener los mensajes nuevos y los añade a la lista que acaba de obtener del fichero.

4.3.1.3.2. Agenda

Esta utilidad proporciona al usuario una gestión del tiempo de forma que los miembros del grupo puedan crear y visualizar los eventos fijados para un grupo.

Hay dos secciones en las que el usuario puede visualizar estos eventos:

Calendario.

El usuario puede ver los asuntos de los eventos de un grupo fijados para un día en concreto, seleccionándolo en el calendario.

Lista de eventos.

Se muestra una lista con todos los eventos disponibles para el grupo actual. El usuario podrá realizar dos operaciones en esta sección:

Ver evento.

Al seleccionar un evento de la lista se muestra al usuario la información detallada del evento, que consta de:

- Asunto.
- Lugar.
- Fecha.
- Hora.

La fecha se mostrará en formato ‘día/mes/año’ y la hora ‘hora:minutos’ en caso de haber sido introducida, en caso contrario, pondrá ‘Sin especificar’

Crear evento.

En esta subsección se permite la creación de un nuevo evento en el que el usuario puede introducir los siguientes campos:

- Asunto.
- Lugar.
- Fecha.
- Hora.

En esta sección no se permitirá guardar eventos que contengan los campos vacíos, a excepción del campo de la hora, que se pondrá un valor por defecto desde el servidor en la base de datos, en caso de no haber sido especificada por el usuario.

Hay una restricción en este módulo que limita la existencia de dos eventos del mismo grupo con la misma descripción para el asunto, permitiendo que puedan existir dos eventos con el mismo asunto para grupos diferentes, debido a que los eventos tienen asignados en la base de datos el identificador al grupo al que pertenecen y así poder facilitar las consultas.

Existe un método en el servidor que va comprobando diariamente las fechas de los eventos guardados para borrar los registros pasados y así liberar memoria.

4.3.1.3.3. Información del grupo

En este módulo, el usuario visualiza la información del grupo, donde encontrará el nombre del grupo, la lista de usuarios que han aceptado la invitación, la opción de añadir más miembros y la de salir del grupo.

El método que se sigue para añadir un nuevo alumno al grupo es el mismo que cuando se crea un grupo nuevo, de manera que se comprueba si el campo está vacío, si el usuario existe y si el usuario no estaba añadido anteriormente a la lista. Si se cumplen las condiciones anteriores, se manda la petición al nuevo miembro.

A la hora de salir del grupo, se borra el archivo del chat perteneciente a ese grupo de la memoria del teléfono, y los mensajes que quedan pendientes de recibir de otros participantes de la base de datos.

En caso de ser el último miembro del grupo se borran de la base de datos todos los registros referentes a ese grupo:

- Registro al grupo.
- Invitaciones a otros usuarios al grupo.

- Mensajes pendientes de envío.
- Eventos.

4.3.1.3.4. Submenú de información de la aplicación

Existe un modulo presente en todas las pantallas excepto en la de identificación al que se entra a través del botón de menú que tienen disponible los dispositivos Android. A continuación se muestran las diferentes opciones de esta sección que el usuario puede escoger:

- Recargar. Este módulo sirve para recargar la pantalla actual, es decir, si se está en una pantalla que obtiene datos del servidor y se pulsa esta opción, se obtendrán de nuevo los datos verificando si ha habido algún cambio en la base de datos en el periodo en el que el cliente ha tenido en primer plano esa ventana.
- Inicio. A través de esta opción el usuario podrá volver a la pantalla principal de la aplicación, la de la lista de grupos, de manera que no tenga que estar volviendo con el botón de retroceder, físico o táctil, dependiendo del terminal, a través de toda la pila de actividades.
- Ayuda. Con esta elección el usuario podrá ver una breve ayuda de la aplicación donde se le explica el funcionamiento básico de la aplicación.
- AcercaDe. En esta última alternativa se le mostrará al usuario la información del grupo de implementación del proyecto compuesto por los desarrolladores del código para diferentes plataformas y el tutor que ha realizado un seguimiento de las fases de implementación del proyecto.

4.3.2 COMUNICACIÓN CLIENTE Y SERVIDOR

En esta sección se explica la metodología que se sigue para que el cliente obtenga los datos que el servidor recibe de la base de datos.

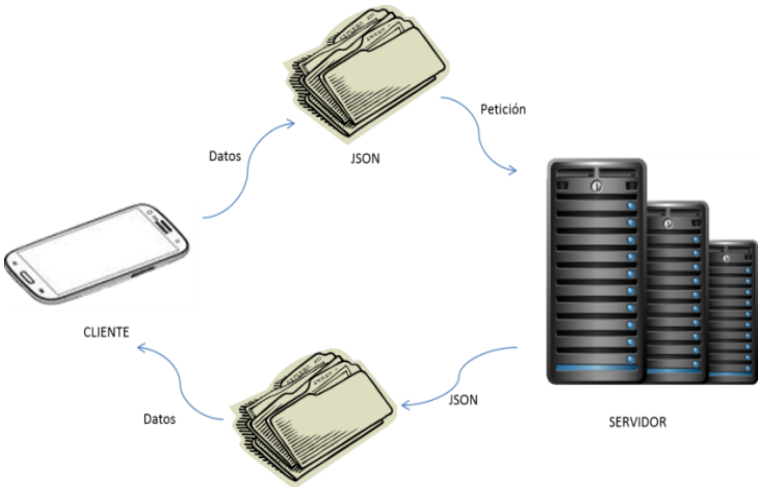


Ilustración 16. Diagrama conexión cliente-servidor

En la *Ilustración 16* se observa el diagrama de la comunicación del cliente con el servidor.

Para realizar las peticiones desde el cliente, se utilizan una serie de comandos diferentes dependiendo de los datos que se quieren obtener. Estos comandos se le envían con una URL que sigue el siguiente formato:

http://"IP del host" : "Puerto del host" / "ruta del archivo del servlet" ? "comando"

Donde la IP (Internet Protocol [12]) y el puerto son parámetros de red correspondientes al servidor, la ruta del archivo es la localización en el sistema del archivo '.java' del servlet y el comando es la acción que se realizará en el servidor.

A continuación se muestran los comandos con los parámetros que se le pasan al servidor y los resultados que devuelve.

4.3.2.1. Comando de identificación

El comando de identificación se denomina 'login' y es el enviado para validar el inicio de sesión en la aplicación.

Los parámetros de entrada son los mostrados en la *Tabla 3*.

Parámetros de Entrada
Username
Password

Tabla 3. Parámetros de Entrada para el comando `LOGIN`

El parámetro de salida que el servidor manda al cliente con el resultado de la consulta a la base de datos es el mostrado en la *Tabla 4*.

Parámetros de Salida
Result

Tabla 4. Parámetros de Salida para el comando `LOGIN`

El resultado será 'OK' si el usuario y contraseña introducidos son correctos, mientras que si ocurre algo en la conexión con la base de datos, el usuario y contraseña no coinciden o el usuario no existe se devuelve como resultado 'NOK'

4.3.2.2. Comando de obtención de grupos registrados

El comando 'getGroups' se manda inmediatamente después de que el cliente obtiene la información de que el login ha sido correcto. Mediante esta instrucción el cliente obtiene la lista de los grupos a los que está subscrito.

En la *Tabla 5* se puede ver el parámetro enviado al servidor para la consulta.

Parámetros de Entrada
Username

Tabla 5. Parámetros de Entrada para el comando 'GETGROUPS'

El servidor devuelve una lista con dos elementos por cada objeto, de forma que el cliente reciba los nombres de los grupos que tiene registrados y el identificador único de cada grupo.

En la Tabla 6 se pueden ver los parámetros mencionados anteriormente.

Parámetros de Salida
groupID
groupName

Tabla 6. Parámetros de Salida para el comando 'GETGROUPS'

4.3.2.3. Comando de obtención de grupos pendientes

La instrucción 'getPetitions' se envía para recibir el listado de las peticiones que un usuario tiene pendientes.

En la *Tabla 7* se muestra el parámetro enviado para ser procesado desde el servidor.

Parámetros de Entrada
Username

Tabla 7. Parámetros de Entrada para el comando 'GETPETITIONS'

El servidor devuelve una lista con dos elementos por cada objeto, de forma que el cliente reciba los nombres de los grupos que el usuario tiene pendientes de aceptación y el identificador único de cada grupo.

En la *Tabla 8* se pueden ver los parámetros mencionados anteriormente.

Parámetros de Salida
groupID
groupName

Tabla 8. Parámetros de Salida para el comando 'GETPETITIONS'

4.3.2.4. Comando de confirmación de petición

El comando 'acceptPetitions' se manda al aceptar una invitación a un grupo pendiente.

En este método, el servidor realiza diversas funciones internamente, es decir, al aceptar la invitación de un nuevo grupo, se debe de añadir este grupo a la lista de grupos asignada para un usuario y borrarlo de la lista de pendientes.

Los parámetros enviados al servidor por parte del cliente se pueden ver en la *Tabla* 9.

Parámetros de Entrada
Username
groupID
groupName

Tabla 9. Parámetros de Entrada para el comando 'ACCEPTPETITIONS'

El parámetro de salida que el servidor manda al cliente con el resultado de la consulta a la base de datos es el mostrado en la *Tabla 10*.

Parámetros de Salida
Result

Tabla 10. Parámetros de Salida para el comando 'ACCEPTPETITIONS'

El resultado será 'OK' si el grupo ha sido aceptado y creado correctamente, mientras que si ocurre algo en la conexión con la base de datos o surge algún problema durante el proceso, se devuelve como resultado 'NOK'

4.3.2.5. Comando de comprobación de peticiones

La instrucción 'checkPetitions' se envía paralelamente a getGroups, es decir, una vez se ha comprobado que el usuario ha sido validado correctamente. Mediante este comando se comprueba si el usuario tiene peticiones pendientes.

En la *Tabla 11* se muestra el parámetro enviado para ser procesado desde el servidor.

Parámetros de Entrada
Username

Tabla 11. Parámetros de Entrada para el comando 'CHECKPETITIONS'

El parámetro de salida que el servidor manda al cliente con el resultado de la consulta a la base de datos es el mostrado en la *Tabla 12*.

Parámetros de Salida
Result

Tabla 12. Parámetros de Salida para el comando 'CHECKPETITIONS'

El resultado será 'OK' si existen grupos pendientes para el usuario, mientras que si ocurre algo en la conexión con la base de datos o surge algún problema durante el proceso, se devuelve como resultado 'NOK'

4.3.2.6. Comando de comprobación de usuario

El comando 'checkPartner' se manda para comprobar la existencia de un alumno en la base de datos. Cada vez que se intenta añadir un miembro nuevo a la lista de alumnos de un grupo se llama a este comando para que realice la comprobación.

Se puede ver el parámetro enviado al servidor para la consulta en la *Tabla 13*:

Parámetros de Entrada
Username

Tabla 13. Parámetros de Entrada para el comando 'CHECKPARTNER'

El parámetro de salida que el servidor manda al cliente con el resultado de la consulta a la base de datos es el mostrado en la *Tabla 14*.

Parámetros de Salida
Result

Tabla 14. Parámetros de Salida para el comando 'CHECKPARTNER'

El resultado será 'OK' si el usuario consultado existe en la base de datos, mientras que si ocurre algo en la conexión con la base de datos, surge algún problema durante el proceso, o el usuario no existe, se devuelve como resultado 'NOK'

4.3.2.7. Comando de creación de nuevo grupo

La instrucción de 'insertGroup' se manda cuando se crea un nuevo grupo haciendo una inserción en la base de datos.

Se muestran en la *Tabla 15* los parámetros enviados para su procesado en el servidor.

Parámetros de Entrada
Username
groupName

Tabla 15. Parámetros de Entrada para el comando 'INSERTGROUP'

El parámetro de salida que el servidor manda al cliente con el resultado de la consulta a la base de datos es el mostrado en la *Tabla 16*.

Parámetros de Salida
groupID

Tabla 16. Parámetros de Salida para el comando 'INSERTGROUP'

El servidor devolverá un identificador único del nuevo grupo creado, que será utilizado posteriormente para las invitaciones a los diversos miembros del grupo.

4.3.2.8. Comando de creación de invitación a grupo

El comando 'insertPetition' se envía tantas veces como miembros tenga un grupo cada vez que se crea uno nuevo. También se envía cada vez que se añade un nuevo miembro desde la pestaña de información del grupo. Por esto es necesario disponer del id del grupo al que se quiere realizar la invitación y en el caso de ser una creación de grupo hay que hacer primeramente una llamada a la instrucción 'insertGroup'.

En la *Tabla 17* se observan los diferentes parámetros enviados desde el cliente.

Parámetros de Entrada
Username
groupID
groupName

Tabla 17. Parámetros de Entrada para el comando 'INSERTPETITION'

El parámetro de salida que el servidor manda al cliente con el resultado de la consulta a la base de datos es el mostrado en la *Tabla 18*.

Parámetros de Salida
Result

Tabla 18. Parámetros de Salida para el comando 'INSERTPETITION'

El resultado será 'OK' si las invitaciones se han realizado correctamente, mientras que si ocurre algo en la conexión con la base de datos o surge algún problema durante el proceso se devuelve como resultado 'NOK'

4.3.2.9. Comando de obtención de alumnos de un grupo

El comando 'getAlumnos' se manda cuando se requiere cargar la información del grupo y mostrar la lista de alumnos pertenecientes a un grupo y que hayan aceptado la solicitud.

En la *Tabla 19* se puede ver el parámetro enviado al servidor para la consulta.

Parámetros de Entrada
groupID

Tabla 19. Parámetros de Entrada para el comando 'GETALUMNOS'

El servidor devuelve una lista de elementos que contendrá los alumnos correspondientes al grupo con el ID de la búsqueda como se puede ver en la *Tabla 20*.

Parámetros de Salida
Username

Tabla 20. Parámetros de Salida para el comando 'GETALUMNOS'

4.3.2.10. Comando de creación de evento

La instrucción 'insertAgenda' se lleva a cabo cuando se genera un nuevo evento para un grupo determinado.

En la *Tabla 21* se pueden observar los parámetros que mandan con este comando.

Parámetros de Entrada
groupID
Lugar
Asunto
Fecha
Hora (Opcional)

abla 21. Parámetros de Entrada para el comando 'INSERTAGENDA'

Se puede observar que hay un campo denominado 'Hora' que es opcional. Esto se debe a que el usuario no tiene la obligación de introducir una hora determinada para el evento que desea crear y porque dependiendo del SO en el que esté implementada la aplicación, se utiliza o no.

Para introducir el evento en la base de datos, se constata si el paquete enviado desde el servidor contiene información para el parámetro de la hora y si no es así, se establece un valor por defecto en la base de datos.

El parámetro de salida que el servidor manda al cliente con el resultado de la consulta a la base de datos es el mostrado en la *Tabla 22*.

Parámetros de Salida
Result

Tabla 22. Parámetros de Salida para el comando 'INSERTAGENDA'

El resultado será 'OK' si la creación del evento se ha realizado correctamente, mientras que si ocurre algo en la conexión con la base de datos, surge algún problema durante el proceso o el evento ya existe con el mismo asunto, se devuelve como resultado 'NOK'

4.3.2.11. Comando de obtención de eventos

El comando 'getAgenda' se envía en cuanto se entra en la subsección de 'Eventos' de la pestaña de Agenda para obtener los eventos correspondientes al grupo actual.

En la *Tabla 23* se muestra el parámetro enviado para obtener dicha información.

Parámetros de Entrada
groupID

Tabla 23. Parámetros de Entrada para el comando 'GETAGENDA'

El servidor devuelve una lista de elementos que contendrá los asuntos de los eventos correspondientes al grupo con el ID de la búsqueda como se puede ver en la *Tabla 24*.

Parámetros de Salida
Asunto

Tabla 24. Parámetros de Salida para el comando 'GETAGENDA'

4.3.2.12. Comando de obtención de evento para un día

La instrucción 'getDateAgenda' se realizará cada vez que el usuario seleccione un día en la pestaña calendario.

En la *Tabla 25* se ven los campos que se mandan desde el cliente para su posterior uso en el servidor.

Parámetros de Entrada
groupID
Fecha

Tabla 25. Parámetros de Entrada para el comando 'GETDATEAGENDA'

El servidor devuelve una lista de elementos que contendrá los asuntos de los eventos correspondientes a la fecha seleccionada y al grupo con el ID de la búsqueda como se puede ver en la *Tabla 26*.

Parámetros de Salida
Asunto

Tabla 26. Parámetros de Salida para el comando 'GETDATEAGENDA'

4.3.2.13. Comando de obtención de información de evento

El comando 'getQuedada' se ejecuta en el momento en que se selecciona un evento de la lista de la sección de eventos.

Los parámetros enviados desde el cliente, destinados al servidor son los mostrados en la *Tabla 27*.

Parámetros de Entrada
groupID
Asunto

Tabla 27. Parámetros de Entrada para el comando 'GETQUEDADA'

El servidor devuelve una serie de campos, mostrados en la *Tabla 28*, que serán los que se muestren en la información de un evento concreto.

En este caso se devuelve un campo opcional, como en el caso de inserción de un evento, que es el parámetro de hora. En este caso el cliente es el que decide si utiliza o no, esa información devuelta por el servidor dependiendo del SO utilizado.

Parámetros de Salida
Lugar
Asunto
Fecha
Hora (Opcional)

Tabla 28. Parámetros de Salida para el comando 'GETQUEDADA'

4.3.2.14. Comando de salida de grupo

El parámetro 'leftGroup' se envía al servidor cuando un miembro quiere salir del grupo.

Esta acción implica que se borren todos los registros correspondientes a ese alumno, excepto los mensajes del chat que queden pendientes de enviar a otros miembros.

En la *Tabla 29* se observan los campos enviados para esta operación.

Parámetros de Entrada
groupID
Username

Tabla 29. Parámetros de Entrada para el comando 'LEFTGROUP'

El servidor borrará todos los registros de las tablas de la base de datos en los que aparezca esa combinación de campos excepto en la tabla del chat donde el Username sea el origen del mensaje.

En el caso de que sea el último miembro del grupo, se borrarán todos los registros de las tablas de la base de datos donde aparezca el ID del grupo, para optimizar el almacenamiento de la información y no mantener registros innecesarios, que no se van a consultar más veces.

Finalizado este proceso de limpieza de datos, el servidor contesta al servidor con un parámetro mostrado en la *Tabla 30*.

Parámetros de Salida
Result

Tabla 30. Parámetros de Salida para el comando 'LEFTGROUP'

El resultado será 'OK' si la salida del grupo se ha realizado correctamente, mientras que si ocurre algo en la conexión con la base de datos o surge algún problema durante el proceso, se devuelve como resultado 'NOK'

4.3.2.15. Comando de obtención de mensajes

El comando 'getChat' se efectúa para obtener todos los chats pendientes para un usuario.

En la *Tabla 31* se pueden observar los campos enviados necesarios para la consulta en el servidor.

Parámetros de Entrada
groupID
Username

Tabla 31. Parámetros de Entrada para el comando 'GETCHAT'

El servidor devuelve una serie de campos, mostrados en la *Tabla 32*, que serán los que se muestren en la sección de chat en el orden en que fueron enviados por los demás usuarios.

Parámetros de Salida
Username
Destino
Fecha
Mensaje

Tabla 32. Parámetros de Salida para el comando 'GETCHAT'

4.3.2.16. Comando de creación de mensaje pendiente

La operación de 'insertChat' se realiza cada vez que un usuario envía un mensaje al resto de miembros.

Esta operación se realiza tantas veces como miembros hay en el grupo, es decir, también se inserta un registro para el usuario que envía el mensaje, que se recuperará inmediatamente ya que esta operación hace una llamada interna desde el servidor al comando 'getChat'.

En la *Tabla 33* se observan los parámetros destinados al envío de un mensaje.

Parámetros de Entrada
groupID
Username
Destino
Fecha
Mensaje

Tabla 33. Parámetros de Entrada para el comando 'INSERTCHAT'

El parámetro de salida que el servidor manda al cliente con el resultado de la consulta a la base de datos es el mostrado en la *Tabla 34*.

Parámetros de Salida
Result

Tabla 34. Parámetros de Salida para el comando 'INSERTCHAT'

El resultado será 'OK' si el envío de los mensajes a los miembros del grupo se ha realizado correctamente, mientras que si ocurre algo en la conexión con la base de datos o surge algún problema durante el proceso, se devuelve como resultado 'NOK'.

Un posible ejemplo de comunicación es el mostrado en la *Ilustración 17*.

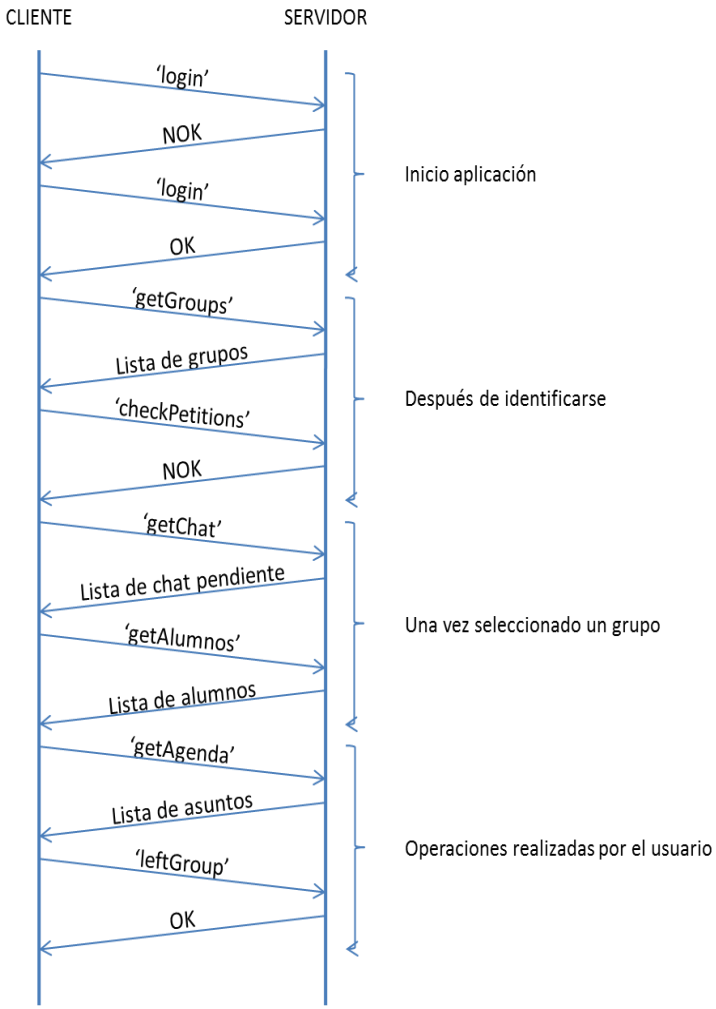


Ilustración 17. Ejemplo de comunicación cliente-servidor

Una vez se ha introducido el comando que se necesita para obtener una información determinada, se procede a encapsular la información que el servidor requiere para realizar la consulta. Estos datos se almacenan en un objeto de tipo JSON que serán enviados posteriormente al servidor.

Para enviarlos al servidor se introducen en un objeto de tipo HTTP (HyperText Transfer Protocol [10]) que mandara la información de forma oculta para evitar posibles intrusiones en el sistema.

A continuación, el cliente espera una respuesta del servidor con los datos obtenidos de la consulta.

Esta información nuevamente es un objeto JSON (puede ser un objeto JSON o un JSONArray, dependiendo del tipo de datos) de donde se obtendrán los datos necesarios para procesar la operación pertinente y cerrar la conexión con el servidor.

4.3.3 ESTRUCTURA DEL SERVIDOR

En este apartado se realiza una explicación detallada del funcionamiento del servidor.

El principal cometido de este bloque es hacer de conexión entre el servidor y la base de datos. Para ello, el cliente realiza una petición que el servidor procesará y posteriormente se realizarán las consultas en la base de datos.

Se pueden distinguir dos tipos de peticiones en la arquitectura de un servidor: peticiones de tipo GET o peticiones de tipo POST.

El 'GET' manda los datos añadiéndolos a la URL de la petición, mientras que el 'POST' los envía en la cabecera. En este caso, se usa el parámetro 'POST' para las peticiones del cliente, ya que por seguridad de los registros de la base de datos es mejor esta instrucción, ya que el 'GET' manda los datos en la URL, siendo visible la instrucción y haciendo así vulnerable la aplicación a posibles modificaciones de los registros, desestabilizando la base de datos.

Las diferentes peticiones que se pueden hacer al servidor son las mostradas en la *Tabla 35*, donde se observan los comandos que envía el cliente y el nombre asignado en el servidor para su procesado:

Comando enviado desde el cliente	Cadena para el procesado en el servidor
test	CMD_TEST
login	CMD_LOGIN
getGroups	CMD_GET_GROUPS
getPetitions	CMD_GET_PETITIONS
acceptPetitions	CMD_ACCEPT_PETITIONS
checkPetitions	CMD_CHECK_PETITIONS
checkPartner	CMD_CHECK_PARTNER
insertGroup	CMD_INSERT_GROUP
insertPetition	CMD_INSERT_PETITION
getAlumnos	CMD_GET_AlUMNOS
insertAgenda	CMD_INSERT_AGENDA
getAgenda	CMD_GET_AGENDA
getDateAgenda	CMD_GET_DATE_AGENDA
getQuedada	CMD_GET_QUEDADA
leftGroup	CMD_LEFT_GROUP
getChat	CMD_GET_CHAT
insertChat	CMD_INSERT_CHAT

Tabla 35. Comandos de las peticiones al servidor

Los comandos que el cliente manda deben de tener el nombre exactamente igual que en el procesado de la URL en el servidor, ya que si no es así, la instrucción no será válida. Es por ello, que en el servidor, a los comandos se les asigna una cadena para que la gestión de peticiones sea más cómoda.

Existe un parámetro que no es consultado desde el cliente denominado 'test'. Este comando sirve para comprobar que la conexión con el servidor está activa. Se puede controlar el estado de la conexión desde cualquier navegador introduciendo esta instrucción en la URL. Esta petición se realiza a través de una petición tipo GET ya que únicamente es una operación de control de la conexión, que no recibirá ningún parámetro y por lo tanto se procesará en el método 'doGet' del servidor.

El resto de comandos existentes se ejecutarán mediante una petición de tipo 'POST', que se procesará en el método 'doPost' del servidor y se hará un tratamiento de la cadena recibida del cliente, para obtener la petición requerida por el usuario.

Una vez que se conoce qué petición se ha realizado, se procede a obtener los datos enviados desde el cliente. Éstos son recibidos encapsulados en un paquete que se guarda en un objeto JSON para procesarlo posteriormente en el servidor.

Cuando se ha finalizado la formación del paquete, se hacen las consultas pertinentes en la base de datos para recuperar los registros deseados.

Según qué tipo de datos se obtengan, los datos recuperados serán procesados de manera distinta, por lo que se distinguen dos formas de preparar los datos para devolverlos al cliente:

- Objeto JSON: En caso de que la operación solo devuelva un parámetro, éste se le devolverá al cliente mediante un objeto de tipo JSON.
- JSONArray: Si el resultado del procesado del comando es una lista de objetos, el servidor devolverá un objeto de tipo JSONArray donde almacenará cada tipo de dato, tanto si es una lista de datos (como puede ser la lista de asuntos de los eventos), como si se trata de una lista de objetos que contienen diferentes datos cada uno (como puede ser la lista de grupos con su correspondiente identificador).

En algunos comandos, el servidor se encarga de realizar operaciones paralelas que son necesarias para determinadas consultas, como comprobaciones de recepción correcta de los parámetros, obtener datos necesarios para la consulta de otras tablas, etc. Algunos ejemplos de estas operaciones pueden ser:

- En el caso de introducir un mensaje de chat sin la información de la fecha y la hora en que fue enviado el texto, el servidor obtiene la fecha y la hora del sistema dándole el mismo formato que si se le mandase desde el cliente y la introduce en la base de datos.
- El servidor hace un procesado distinto tanto si se crea un evento con una hora establecida, como si no de manera que no haya errores en la inserción en la base de datos.

Otro punto a destacar del servidor, es que se ha generado un hilo de ejecución paralelo a la gestión de peticiones donde se borran los registros de eventos pasados en la base de datos. Este método se ejecuta en el momento en el que el servidor empieza a funcionar y se realiza este control diariamente para garantizar un buen control de los registros de la base de datos.

Para ello, hay un método que comprueba si los registros de la tabla que contiene los eventos, tienen la fecha anterior a la actual y si es así lo borra.

4.3.4 COMUNICACIÓN SERVIDOR – BASE DE DATOS

En esta sección se explica el proceso que se sigue para que el servidor obtenga la información de la base de datos.

En la *Ilustración 18* se muestra el diagrama de conexión del servidor con la base de datos.

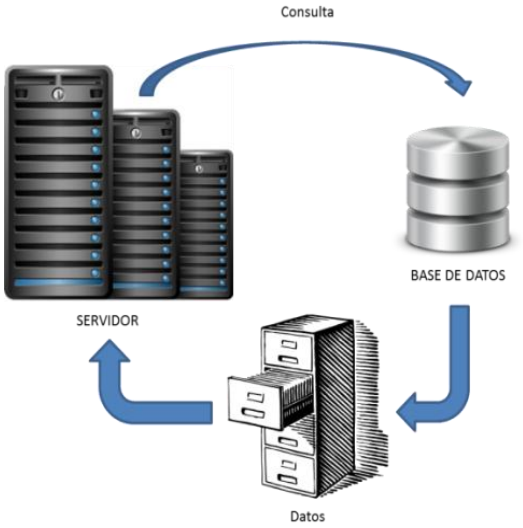


Ilustración 18. Ejemplo de comunicación cliente-servidor

Una vez se ha procesado la consulta del cliente, el servidor realiza una serie de consultas a la base de datos para obtener los datos requeridos por el cliente.

Las consultas utilizadas para la obtención de los registros se basan en sentencias SQL.

Dependiendo del tipo de consulta se utilizan una función u otra, es decir, se dispone de una función 'executeQuery' que se utiliza para realizar consultas que obtengan registros de la base de datos, mientras que se utiliza la función 'updateQuery' en caso de querer modificar algún registro.

A continuación se detallan las diferentes sentencias que se utilizan dependiendo de la operación que se va a realizar en la aplicación.

- INSERT: Esta orden se utiliza para realizar inserciones en las tablas de la base de datos. A la hora de insertar un registro se deberán meter todos los datos, ya

que en caso de estar alguno vacío, en el campo de la tabla se guarda 'NULL' (en español nulo) si no tiene un valor por defecto como ocurre en el campo de 'hora' en la tabla de gestión de eventos.

- DELETE: Esta instrucción sirve para borrar registros de una tabla de la base de datos. Se pueden borrar tanto individualmente como en grupos dentro de una misma tabla. Se le indica una referencia de una columna de la base de datos y se borran todos los registros en lo que sus datos coincidan con esa condición.
- SELECT: Este comando se utiliza para obtener registros de la base de datos. Estos registros no se borran al hacer la consulta, si no que la operación que se realiza es de lectura, manteniendo los valores de los registros.

4.3.5 ESTRUCTURA DE LA BASE DE DATOS

La organización de la base de datos sigue una disposición de cinco tablas que se explican detalladamente a continuación.

4.3.5.1. Gestión de Usuarios

Para la gestión de usuarios se decide crear una tabla denominada 'USERS' donde se establece la disposición de los campos mostrada en la *Tabla 36*.

Columna	Tipo de datos
Id	INTEGER
Name	CHAR
Pass	CHAR

Tabla 36. Tabla 'USERS' de Base de Datos

Contiene los datos de inicio de sesión de los alumnos, que serán el número de identificación personal del alumno de la Universidad Carlos III de Madrid y su contraseña de Campus Global.

Es consultada en dos momentos de la aplicación:

- En el inicio de la sesión, en el momento en que se arranca la aplicación y se quiere acceder a ella, donde se comprueba que el alumno existe y la contraseña coincide.
- Cuando se añade un nuevo miembro a un grupo, dado que se realiza una comprobación de que ese alumno existe, para que no se generen registros en la base de datos que nunca van a ser recuperados.

Esta tabla no se va a modificar nunca desde la aplicación, ya que es donde se encuentran todos los datos del usuario y son personales para el resto de usuarios. Por lo tanto es una tabla que se consultará desde el servidor, a la que los alumnos no tendrán acceso.

4.3.5.2. Gestión de Grupos

Para la gestión de los grupos de la aplicación se dispone de dos tablas diferentes, una para los grupos a los que pertenecen los alumnos y otra con las invitaciones a nuevos grupos.

- Existe una tabla 'GROUPS' que sigue la estructura mostrada en la *Tabla 37*.

Columna	Tipo de datos
Id	INTEGER
Name	CHAR
User	CHAR

Tabla 37. Tabla 'GROUP' de Base de Datos

Esta tabla contiene la vinculación de un alumno con los diferentes grupos en los que está registrado.

Se guarda un registro por cada relación alumno grupo, es decir, por cada miembro de un grupo existirá un registro, o bien, un registro por cada grupo en que un alumno esté adscrito.

Una característica de esta tabla es que el identificador introducido es único para cada grupo. El procedimiento que asigna este valor se realiza en el servidor, el cual consulta en la base de datos cual es ID del último registro y lo incrementa en una unidad. De esta manera, un usuario podrá disponer de varios grupos con diferentes nombres. Esto se debe a que cuando un usuario crea un grupo, no sabe los nombres de los grupos en los que están unidos el resto de compañeros a los que se quiere invitar y así no surgirán problemas de referencias con los nombres y se hace el procesado correctamente.

La gran parte de las gestiones de la base de datos se fundamentan en el identificador de grupo, ya que la mayoría de las consultas se realizan a partir de él.

Los demás campos que contiene esta tabla son el nombre del grupo y el usuario correspondiente a la columna 'Name' de la tabla 'USERS'.

La tabla 'GROUP' es consultada desde varios puntos de la aplicación:

- Cada vez que se muestra la pantalla de la lista de los grupos registrados de un alumno, para cargar la lista.
- Cada vez que se crea un nuevo grupo, para obtener el nuevo identificador.
- Cada vez que se quiere salir de un grupo, para borrar el registro.

- La otra tabla que se utiliza para la gestión de grupos, es la tabla denominada 'PENDIENTES', que sigue la disposición de los campos que se ve en la *Tabla 38*.

Columna	Tipo de datos
Id_Grupo	INTEGER
Grupo	CHAR
User	CHAR

Tabla 38. Tabla 'PENDIENTES' de Base de Datos

Contiene las invitaciones a los grupos que los alumnos tienen pendientes.

En esta tabla se almacena el identificador del grupo del que procede la invitación, el nombre del grupo y el usuario al que va dirigida la petición.

Podría parecer innecesario el haber introducido el nombre del grupo en la tabla, ya que se podría consultar en la tabla de 'GROUPS', pero se ha tomado la decisión de incluirlo para reducir el coste computacional que supone el tener que buscar en la tabla de 'GROUPS' el nombre del grupo.

La tabla 'PENDIENTES' es consultada en varios puntos de la aplicación:

- Consultar si existen nuevas invitaciones pendientes en la pantalla de grupos.
- Cargar la lista de invitaciones a nuevos grupos
- En la creación de un nuevo grupo, al realizar invitaciones a los miembros.
- Al visualizar la información de un grupo e insertar un nuevo miembro posteriormente a la creación del grupo.
- Cuando un usuario sale de un grupo y es el último, se borran todas las invitaciones pendientes de ese grupo para optimizar la base de datos.

4.3.5.3. Gestión de eventos

En la *Tabla 39* se muestra la estructura que sigue la tabla que se utiliza para la gestión de eventos llamada 'AGENDA'.

Columna	Tipo de datos
Id_Grupo	INTEGER
Lugar	CHAR
Asunto	CHAR
Fecha	CHAR
Hora	CHAR

Tabla 39. Tabla 'AGENDA' de Base de Datos

Contiene los eventos que los usuarios crean desde la aplicación.

En esta tabla se almacena el identificador del grupo al que corresponde el evento, el lugar del evento, el asunto, la fecha establecida y la hora.

Esta tabla no tiene un identificador único, ya que la gestión de eventos se hace desde el cliente. Se limita la inserción de eventos con el mismo asunto, de manera, que no pueden estar repetidos.

El campo de la hora es opcional a la hora de insertar un nuevo registro. En caso de no recibir este dato, en la inserción se guarda un registro en el que la hora queda definida como 'Sin especificar'.

La tabla 'AGENDA' es consultada desde varios puntos de la aplicación:

- Consultar si existen eventos para un día determinado seleccionado en la pestaña de calendario.
- Cargar la lista con los nombres de los asuntos de los eventos.
- Mostrar la información de un evento determinado
- Crear un evento.
- Cuando un usuario sale de un grupo y es el último, se borran todos los eventos de ese grupo.
- Mientras el servidor esté disponible, cada día se hace una actualización borrando los eventos pasados.

4.3.5.4. Gestión de mensajes de chat

En la *Tabla 40* se puede ver la disposición que sigue la tabla 'CHAT', denominada así la tabla que se usa para almacenar los mensajes pendientes de envío a los diferentes usuarios.

Columna	Tipo de datos
Id_Grupo	INTEGER
user	CHAR
Destino	CHAR
Fecha	CHAR
Mensaje	CHAR

Tabla 40. Tabla 'CHAT' de Base de Datos

Contiene todos los mensajes que están en cola de envío a los usuarios de un grupo.

En esta tabla se almacena el identificador del grupo del que procede el mensaje, el usuario que manda el mensaje, el destinatario, la fecha y el mensaje enviado.

La tabla 'CHAT es consultada desde varios puntos de la aplicación:

- Consultar si existen nuevos mensajes tanto cada vez que se entra, como con la actualización del chat que se hace periódicamente mientras está la pestaña visible.
- Enviar un mensaje nuevo.

CAPÍTULO 5. PRUEBAS

A continuación se muestra los dispositivos utilizados para la realización de las pruebas pertinentes de la aplicación y algunas capturas de las funcionalidades básicas de la aplicación.

5.1 DISPOSITIVOS DE PRUEBA

Los dispositivos que se han utilizado para la realización de las pruebas son los siguientes:

Smartphones:

- Samsung Galaxy mini GT-S5570
  - Versión de Android 2.3.6
  - Pantalla QVGA con una resolución de 240x320 píxeles de 3,14”
  - Memoria interna de 164 MB expandida mediante tarjeta MicroSD de 8GB
  - Procesador ARMv6 con chip Qualcomm MSM7227 a 600MHz
  - 279 MB de memoria RAM
- Sony Xperia U
  - Versión de Android 2.3
  - Pantalla qHD con una resolución de 540x960 píxeles de 3,5”
  - Memoria interna de 8 GB expandida mediante tarjeta MicroSD de 8GB
  - Procesador de doble núcleo STE U8500 a 1 GHz
  - 512 MB de memoria RAM
- Samsung Galaxy S3 GT-I9300
  - Versión de Android 4.1.2
  - Pantalla HD Super Amoled con una resolución de 1280x720 píxeles de 4,8”
  - Memoria interna de 16 GB expandida mediante tarjeta MicroSD de 16GB

- Procesador Quad-core Cortex-A9 con chip Exynos 4412 Quad a 1.4GHz
- 1GB de memoria RAM

Tablet:

- Asus transformer tf101
  - Versión de Android 4.0.3
  - Pantalla WXGA con una resolución de 1280x800 píxeles de 10,1”
  - Memoria interna de 16 GB expandida mediante dos tarjetas MicroSD de 8GB
  - Procesador Dual-core Cortex-A9 con chip Nvidia Tegra 2 T20 a 1 GHz
  - 1GB de memoria RAM

Servidor:

- VirtualBox: Maquina virtual con las siguientes características:
  - Versión 2.4.8
  - SO simulado: Ubuntu 12.04
  - Memoria base: 1024
  - Memoria de video 64 MB
  - Espacio de disco dedicado: 10 GB

5.2 DESARROLLO DE LAS PRUEBAS

Primeramente el desarrollador realiza una serie de pruebas en su terminal de manera que comprueba que los requisitos se cumplen y diversas pruebas de funcionamiento para comprobar todo el sistema de forma que observa que las consultas a la base de datos se realizan correctamente, al igual que la conexión con el servido.

En este paso es cuando se abre un navegador web para comprobar que el servidor está activo. Para ello en la barra de direcciones se introduce el siguiente comando:

http://XXX.XXX.XXX.XXX:YYYY/directorio?test

Donde la serie de ‘X’ es la IP donde se encuentra instalado el servidor, la cadena de ‘Y’ es el puerto, el directorio es la carpeta donde se encuentran las clases donde se encuentra el código fuente del servidor, y el comando ‘test’ es la instrucción necesaria para comprobar el estado de la conexión devolviendo una cadena en el navegador que pone ‘CONEXIÓN OK’ en caso de estar activa o un error de conexión en caso de estar caída.

Posteriormente se realiza la ejecución de las pruebas desde diversos dispositivos de manera que se compruebe la interfaz para las diferentes resoluciones de pantalla. Esto se realiza por un conjunto de usuarios de manera que la búsqueda de posibles fallos sea lo más eficiente posible, al ser una exploración mas exhaustiva al realizarse entre varias personas.

En la *Ilustración 19* se muestra la ejecución en los diferentes dispositivos de pruebas.

Ilustración 19. Ejecución en diferentes terminales

El desarrollador entrega la documentación de la aplicación con los requisitos que la aplicación debía cumplir a los usuarios que van a realizar las pruebas, de manera que esas pruebas se hagan siempre como mínimo. En alguna prueba puede que el usuario encuentre algún error que se le haya pasado al desarrollador o alguna modificación que haría para un mejor funcionamiento, por lo tanto lo redactara en el informe que le devolverá al programador para que solucione el fallo o para que estudie si la posible mejora no conlleva una implementación compleja.

A medida que los usuarios realizan las pruebas de la aplicación, el desarrollador observa el flujo de datos que se está generando en el servidor a través de la base de datos, viendo como se generan registros en las tablas.

5.3 CAPTURAS DE PANTALLA Y EJEMPLOS

A continuación se muestran algunas capturas de pantalla de la aplicación y diferentes requisitos que la aplicación debía cumplir.

5.3.1. CONEXIÓN CON EL SERVIDOR

Se comprueba desde un navegador web que existe conexión con el servidor. En la *Ilustración 20* se puede observar una captura de pantalla de un ordenador realizando la consulta mediante un navegador web.

Ilustración 20. Consulta de estado de conexión

5.3.2. PANTALLA INICIAL DE IDENTIFICACIÓN

En caso de que el usuario no introduzca alguno de los campos, se mostraran los mensajes de error correspondientes que se muestran en la *Ilustración 21*.

Ilustración 21. Captura de la sección de identificación

5.3.3. PANTALLA DE LA LISTA DE GRUPOS

Se comprueba que se visualizan los grupos y que se actualiza la descripción del botón de grupos pendientes. En la *Ilustración 22* se visualizan las pruebas mencionadas anteriormente.

Ilustración 22. Captura de la sección de lista de grupos registrados

5.3.4. PANTALLA DE GRUPOS PENDIENTES

En esta sección se comprueba que el usuario recibe correctamente las invitaciones a grupos y que se pueden aceptar correctamente. En la *Ilustración 23* se muestran los resultados de estas pruebas.

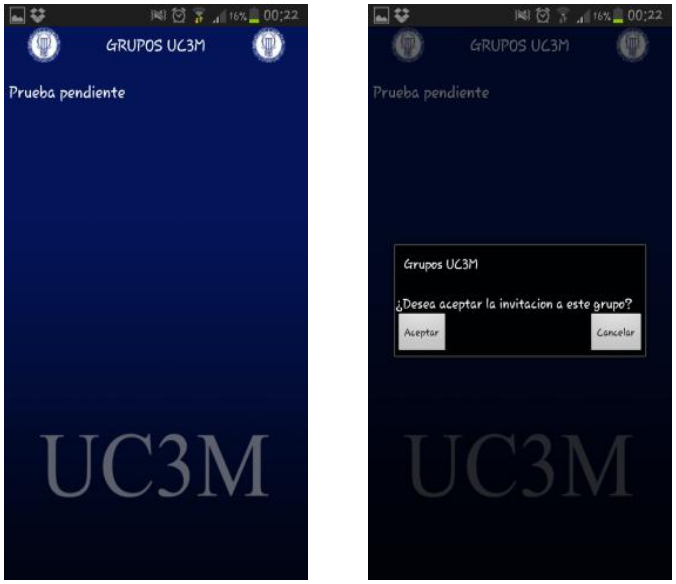


Ilustración 23. Captura de la sección de lista de grupos pendientes

5.3.5. PANTALLA DE CREACIÓN DE GRUPOS

Las pruebas realizadas para este bloque son la correcta inserción de los campos necesarios y que la creación del grupo se realiza correctamente. A continuación se pueden ver algunas capturas en la *Ilustración 24*.

Ilustración 24. Captura de la sección de creación de un grupo

5.3.6. PANTALLA DE GRUPO

En esta ventana se prueban los tres módulos diferentes correspondientes a un grupo determinado.

5.3.6.1. Chat

Se comprueba que se reciben correctamente los mensajes pendientes, que se envían correctamente y que se cargan los mensajes guardados en la memoria del teléfono. En la *Ilustración 25* se muestran unas capturas de las pruebas citadas.

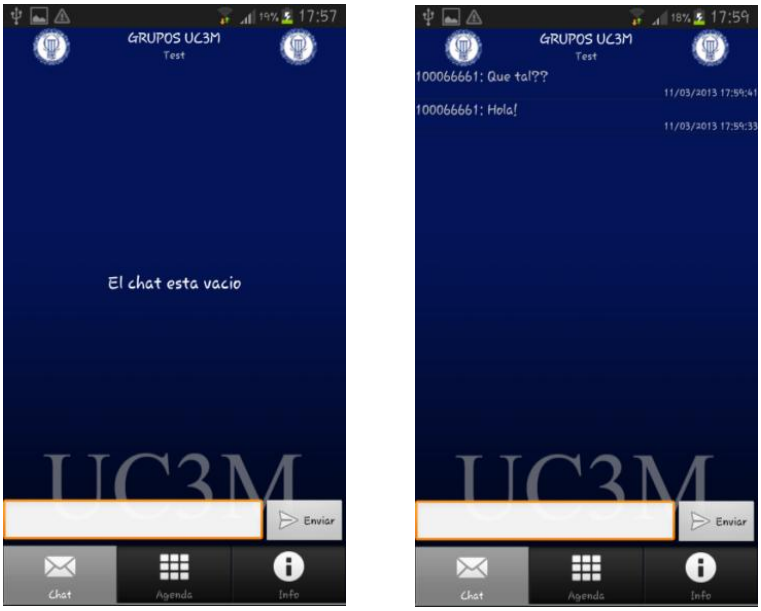


Ilustración 25. Captura de la sección de mensajes de chat

5.3.6.2. Agenda

En este modulo se examina la realización correcta de las operaciones con eventos en las dos diferentes secciones.

- Se verifica que se muestra el día actual correctamente y se ven los asuntos de los eventos correspondientes al día seleccionado.
- Se comprueba que se ve la lista de los asuntos establecidos para los eventos de un grupo, que se visualiza la información de un grupo determinado y que se crean correctamente los grupos verificando que los campos se introducen correctamente.

En la *Ilustración 26* se pueden ver algunas de las pruebas anteriores.

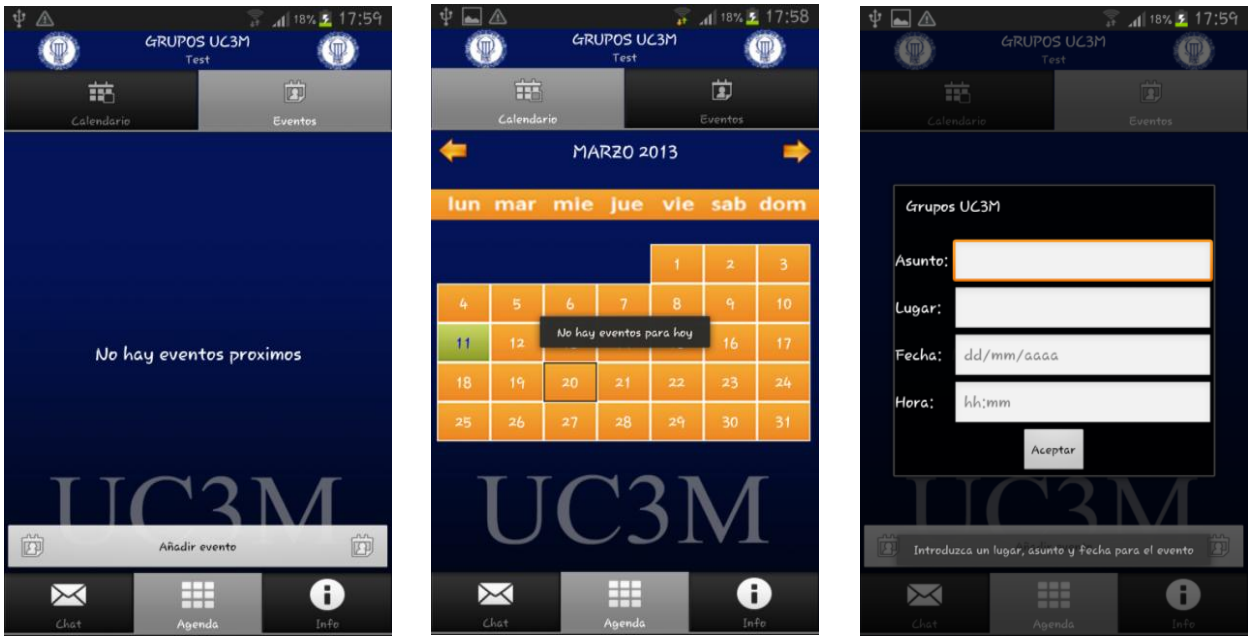


Ilustración 26. Captura de la sección de gestión de eventos

5.3.6.3. Información

En esta sección se prueba que se visualizan correctamente los miembros del grupo, se añaden nuevos miembros y que es posible salir del grupo adecuadamente. A continuación se visualizan alguna de las capturas en la *Ilustración 27*.

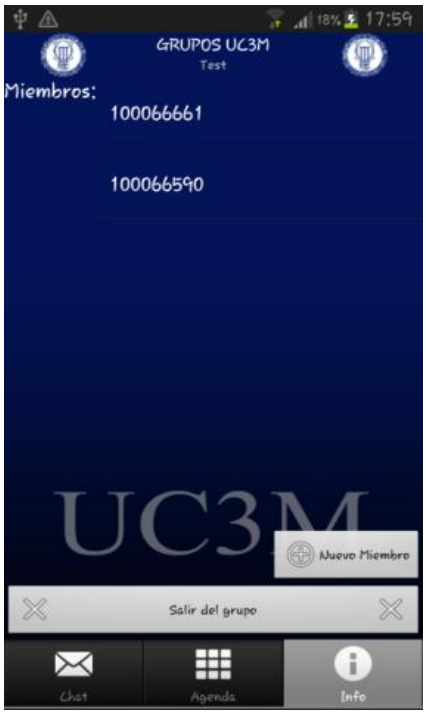


Ilustración 27. Captura de la sección de información

Hay que realizar las pruebas también de que la aparición y funcionamiento del submenú de Android es correcto. En la *Ilustración 28* se muestra el diseño de este menú.

Ilustración 28. Consulta de estado de conexión

En la *Tabla 41* se muestran los requisitos de cada sección de la aplicación que han sido comprobados.

	Prueba a	Prueba b	Prueba c	Prueba d	Prueba e	Prueba f	Prueba g
Identificación	OK	OK	OK	-	-	-	-
Grupos	OK	OK	OK	-	-	-	-
Nuevo grupo	OK	OK	OK	OK	OK	OK	OK
Pendientes	OK	OK	-	-	-	-	-
Grupo	OK	-	-	-	-	-	-
Chat	OK	OK	OK	OK	OK	OK	-
Agenda	OK	OK	OK	OK	OK	OK	OK
Información	OK	OK	OK	-	-	-	-
Submenú	OK	-	-	-	-	-	-

Tabla 41. Cumplimiento de requisitos

Por lo tanto se concluye que, una vez realizadas todas las pruebas de la aplicación, se han cumplido todos los requisitos con éxito.

Aparte de los requisitos iniciales también se ha comprobado el funcionamiento de las mejoras que se consideraron oportunas en la primera ronda de pruebas y que se implementaron posteriormente. Entre ellas se encuentran mejoras de la interfaz, recarga periódica de los mensajes del chat, borrado periódico de eventos antiguos o comprobaciones del estado de la conexión con el servidor.

CAPÍTULO 6. CONCLUSIONES

6.1 INTRODUCCIÓN

Una vez se han cumplido los objetivos y requisitos de la aplicación y se han desarrollado una serie de pruebas de funcionamiento, se puede dar por finalizado este proyecto con un gran resultado que se detalla a continuación.

A pesar de haber obtenido un resultado satisfactorio, se proponen una serie de posibles mejoras que se pueden implementar en un futuro para la aplicación y una serie de consideraciones para el proyecto.

6.2 RESULTADOS

Después de estudiar la viabilidad de los posibles proyectos que se habían planteado, se decidió realizar una aplicación que ofreciese varios servicios a los alumnos de la Universidad Carlos III de Madrid.

La principal pregunta que se realizó para este desarrollo fue: ¿Qué te habría gustado tener durante los años que has estado en la universidad?'. Así surgieron varias ideas de las que posteriormente se hizo una selección de las más funcionales e importantes

Debido a que la realización de la parte de servidor y base de datos del proyecto se ha compartido entre dos alumnos, se decidió añadir cierto grado de complejidad a la parte del cliente implementando varios servicios que serian integrados en la aplicación.

Una vez se finalizó la implementación de la aplicación por ambas partes se procedió a realizar las pruebas pertinentes de la aplicación general y se concluyó que la implementación anterior había obtenido resultados bastante buenos, pero había que corregir ciertos fallos encontrados e implementar algunas mejoras que no se habían tenido en cuenta en un inicio pero posteriormente se han considerado oportunas.

Realizados estos cambios, se procede con la última sesión de pruebas para dar por finalizada la aplicación con resultados exitosos.

En conclusión, la ejecución de este proyecto se ha aprovechado para ampliar conocimientos de programación a otros lenguajes y para aumentar el interés por el desarrollo de aplicaciones móviles. Teniendo en cuenta la situación del mercado en la actualidad, el campo de desarrollo de aplicaciones móviles está en auge, por lo que la realización de este proyecto, supone una ventaja al haber adquirido conocimientos sobre la materia.

6.3 TRABAJO FUTURO

Se han considerado una serie de mejoras de implementación que se podrían implementar en un futuro para siguientes versiones de la aplicación. Algunas de estas mejoras son:

- Implementar un chat instantáneo, que funcione aunque la aplicación esté cerrada. Esta opción se descarto en un inicio por el consumo de batería, pero si los alumnos lo consideran necesario se puede implementar.
- Poder editar la información de los eventos.
- Disponer de información de los grupos a los que te invitan
- Realizar otra mini aplicación integrada dentro del proyecto en donde los alumnos puedan compartir archivos en un grupo.

Además de las mejoras de código se ha estado contactando con el Servicio de Informática de la universidad de forma que la aplicación pase, en un futuro, a formar parte del conjunto de aplicaciones de la universidad, obteniendo un certificado para dar fiabilidad en el mercado de aplicaciones al tener una firma de validez de la universidad. Esto supone que los alumnos puedan introducir sus datos de identificación con seguridad de que la aplicación no es de algún desarrollador que está obteniendo sus datos sin permiso.

Finalmente se ha planteado también la posibilidad de realizar un registro de software de la aplicación, de manera que quede registrado el código del proyecto para su posible distribución a otras universidades u otros centros docentes.

CAPÍTULO 7. PLANIFICACIÓN Y PRESUPUESTO

7.1 PLANIFICACIÓN

A continuación se muestra la duración de cada fase del proyecto desglosando cada tarea en los días laborables que se tarda en realizar cada una.

Primeramente se muestra en la *Tabla 42* el periodo de duración de la primera fase: el estudio previo del proyecto.

Nombre de tarea	Duración	Comienzo	Fin
Fase Bloque Estudio Previo	**10 días**	**lun 01/10/12**	**mar 16/10/12**
Estudio de posible PFG	5 días	lun 01/10/12	vie 05/10/12
Análisis de la viabilidad	5 días	lun 08/10/12	mar 16/10/12

Tabla 42. Periodo bloque estudio previo

A continuación se puede ver el segundo periodo de la aplicación: el bloque de la parte del cliente. En la *Tabla 43* se muestra la planificación para esta parte.

Nombre de tarea	Duración	Comienzo	Fin
Fase Bloque Cliente	**40 días**	**mié 17/10/12**	**lun 17/12/12**
Diseño de la interfaz	5 días	mié 17/10/12	mar 23/10/12
Implementación de la interfaz	30 días	mié 24/10/12	lun 10/12/12
Pruebas de la interfaz	5 días	mar 11/12/12	lun 17/12/12

Tabla 43. Periodo bloque cliente

En la *Tabla 44* se incluyen los días empleados para la realización del bloque perteneciente al servidor y a la base de datos.

Nombre de tarea	Duración	Comienzo	Fin
Fase Bloque Servidor y Base de Datos	**30 días**	**mar 18/12/12**	**vie 08/03/13**
Diseño del servidor	10 días	mar 18/12/12	vie 08/02/13
Diseño de la base de datos	10 días	mar 18/12/12	vie 08/02/13
Implementación del servidor	10 días	lun 11/02/13	vie 22/02/13
Implementación de la base de datos	10 días	lun 11/02/13	vie 22/02/13
Implementación la unión cliente-servidor	5 días	lun 25/02/13	vie 01/03/13
Implementación la unión servidor-base de datos	5 días	lun 04/03/13	vie 08/03/13

Tabla 44. Periodo bloque servidor y BD

Se muestra el desglose de tiempo dedicado para la fase de pruebas y mejoras en la *Tabla 45*.

Nombre de tarea	Duración	Comienzo	Fin
Fase Bloque de Pruebas	**19 días**	**lun 11/03/13**	**lun 15/04/13**
Pruebas de la aplicación	4 días	lun 11/03/13	jue 14/03/13
Arreglos de errores encontrados	5 días	vie 15/03/13	vie 22/03/13
Implementación de mejoras de la aplicación	5 días	mar 02/04/13	lun 08/04/13
Pruebas definitivas	5 días	mar 09/04/13	lun 15/04/13

Tabla 45. Periodo bloque pruebas

Para finalizar se observa en la *Tabla 46* el bloque final perteneciente al desarrollo de la memoria del proyecto.

Nombre de tarea	Duración	Comienzo	Fin
Fase Bloque de Redacción de la Memoria	**30 días**	**mar 16/04/13**	**vie 31/05/13**
Desarrollo de la memoria	30 días	mar 16/04/13	vie 31/05/13

Tabla 46. Periodo primera fase

En conclusión, si se resume el tiempo total del proyecto sumando los días de todos los bloques, obtenemos los datos de la *Tabla 47*.

Nombre de tarea	Duración	Comienzo	Fin
Proyecto de Fin de Grado	**129 días**	**lun 01/10/12**	**vie 31/05/13**
Fase Bloque Estudio Previo	**10 días**	**lun 01/10/12**	**mar 16/10/12**
Fase Bloque Cliente	**40 días**	**mié 17/10/12**	**lun 17/12/12**
Fase Bloque Servidor y Base de Datos	**30 días**	**mar 18/12/12**	**vie 08/03/13**
Fase Bloque de Pruebas	**19 días**	**lun 11/03/13**	**lun 15/04/13**
Fase Bloque de Redacción de la Memoria	**30 días**	**mar 16/04/13**	**vie 31/05/13**

Tabla 47. Desglose temporal de bloques

Para ello se han considerado como no laborables a la hora de la realización del proyecto, debido a los festivos existentes y a los periodos de exámenes, los días mostrados en la *Tabla 48*:

Descripción	Comienzo	Fin
Festivo Local de Leganés	11/10/2012	11/10/2012
El Pilar	12/10/2012	12/10/2012
Todos los Santos	01/11/2012	01/11/2012
Nuestra Señora de la Almudena	09/11/2012	09/11/2012
Día de la Constitución Española	06/12/2012	06/12/2012
Puente de la Constitución Española	07/12/2012	07/12/2012
Inmaculada Concepción	08/12/2012	08/12/2012
Vacaciones de Navidad	24/12/2012	07/01/2013
Preparación de Exámenes	08/01/2013	31/01/2013
San José	18/03/2013	18/03/2013
Vacaciones de Semana Santa	25/03/2013	01/04/2013
Día del trabajador	01/05/2013	01/05/2013
Día de la Comunidad de Madrid	02/05/2013	02/05/2013
Puente Día de la Comunidad de Madrid	03/05/2013	03/05/2013
San Isidro	15/05/2013	15/05/2013

Tabla 48. Días no laborables

7.1.1. DIAGRAMA DE GANTT

En la *Ilustración 29* se puede observar el diagrama de Gantt del proyecto, donde se muestra en una grafica de progreso el desglose de las tareas, realizado anteriormente.

La barra verde superior engloba la totalidad de tareas del proyecto, indicando el periodo temporal total de realización del proyecto.

Las barras naranjas simbolizan la tarea resumen de cada bloque, mientras que las azules son las diferentes tareas que se realizan en el proyecto.

Las flechas que unen las diferentes tareas son necesarias para interpretar las relaciones existentes entre las tareas, es decir, si una tarea depende de la finalización de otra para poder comenzar a realizarse.

Los días visualizados con fondo gris son los días correspondientes a los festivos o periodo vacacional establecido en el apartado anterior. No se visualizan todos los días citados anteriormente debido a la resolución del eje temporal del diagrama de Gantt

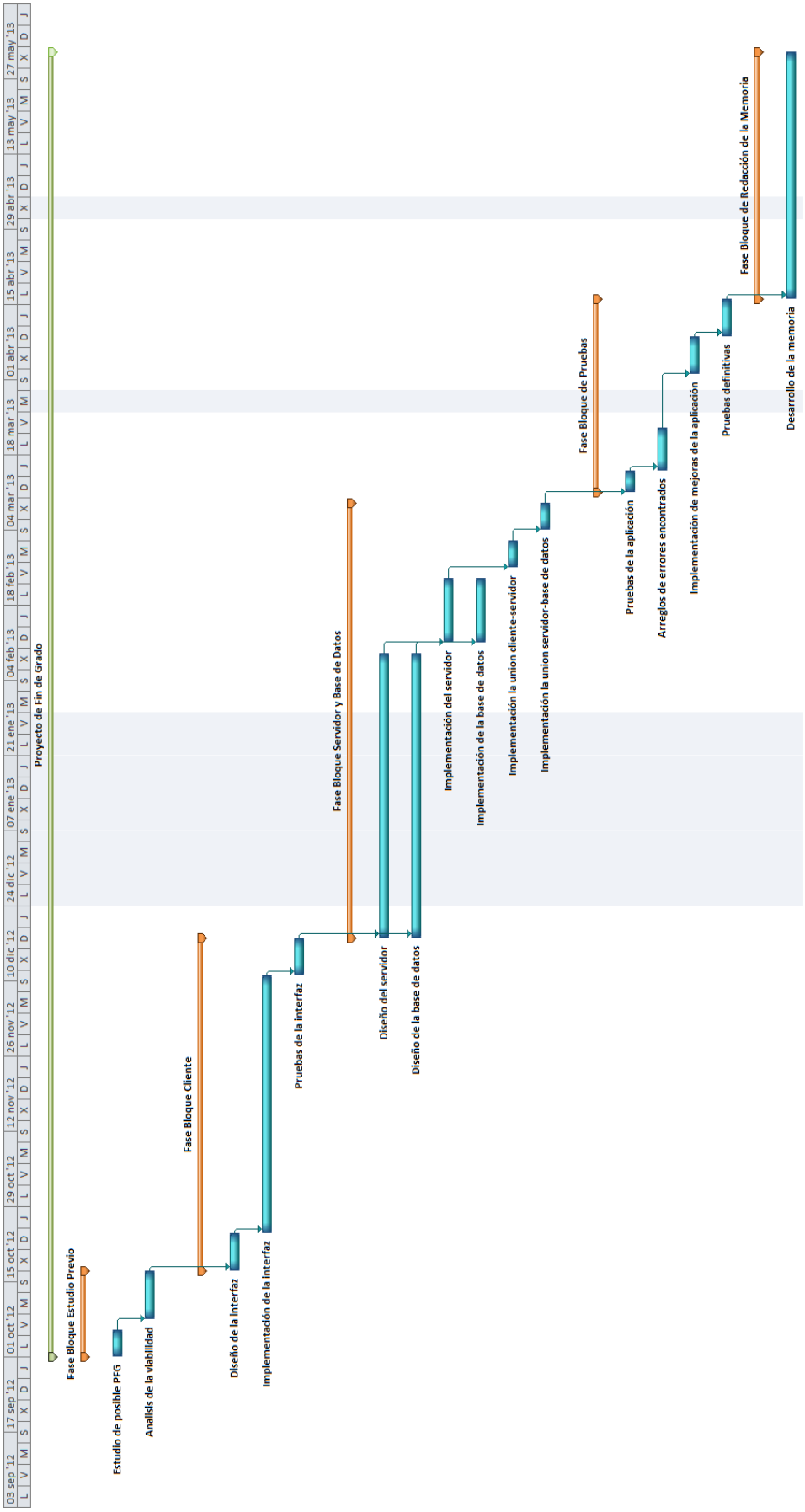


Ilustración 29. Diagrama de Gantt

7.2 PRESUPUESTO

En este apartado se detalla el presupuesto detallado estimado para el proyecto, donde se incluyen los gastos de personal, amortización de equipos, entre otros, siguiendo una guía presupuestaria establecida por la Universidad Carlos III de Madrid [16].

Se establece que un hombre al mes dedica un máximo de 131.25 horas según se establece en el documento citado en este mismo apartado.

Si se establece que se dedican, en media, 3 horas diarias a la realización del proyecto, debido a que no todos los días se ha trabajado y que no se dedica el mismo tiempo, se obtiene que la dedicación de un hombre mes a partir de la siguiente fórmula:

$$Dedicacion(mes) = \frac{Horas\ Dia * Dias\ Totales * Nº\ de\ Hombres}{Máximo\ de\ Horas\ Mensuales}$$

Por lo tanto el resultado de la dedicación de los diferentes trabajadores que han formado parte del proyecto:

- Ingeniero Sénior:

$$Dedicacion(mes) = \frac{3 * (10(estudio\ previo) + 15(avances\ proyecto) + 30(desarrollo\ memoria)) * 1}{131.25} = 1.26\ Hombres\ mes$$

- Ingeniero Junior 1:

$$Dedicacion(mes) = \frac{3 * 129 * 1}{131.25} = 2.95\ Hombres\ mes$$

- Ingeniero Junior 2:

$$Dedicacion(mes) = \frac{3 * (10(estudio\ previo) + 30(implementacion\ servidor\ y\ BD)) * 1}{131.25} = 0.91\ Hombres\ mes$$

- Verificador de pruebas:

$$Dedicacion(mes) = \frac{3 * (4 + 5 + 5) * 3}{131.25} = 0.96\ Hombres\ mes$$

A continuación se realiza un cálculo del coste de personal para el desarrollo del proyecto y se muestra en la *Tabla 49*.

Apellidos y Nombre	Categoría	Dedicación	Coste hombre mes	Coste
Estévez Ayres, Iria Manuela	Ingeniero Sénior	1,26	4.289,54€	5.404,82€
Argüeso Gonzáles, David	Ingeniero Junior 1	2,95	2.694,39€	7.948,45€
Pozo Santiago, Juan Manuel	Ingeniero Junior 2	0,91	2.694,39€	2.451,89€
Sin especificar	Verificadores de pruebas [17]	0,96	2.086,41€	2.002,95€
	TOTALES:	**6,08**	**11.764,73 €**	**17.808,12€**

Tabla 49. Coste del personal

Para calcular el coste que conllevará la contratación del personal para el desarrollo de este proyecto se sigue la siguiente ecuación:

$$Coste\ de\ desarrollador\ (€) = Salario\ al\ mes\left(\frac{€}{mes}\right) * Dedicacion(mes)$$

Para calcular el coste de la amortización de los medios utilizados para el proyecto se utiliza la siguiente fórmula:

Pero primeramente, se establece el valor bruto de los medios utilizados donde se puede observar los diferentes valores del IVA correspondientes a los años 2012 y 2013 en la *Tabla 50*:

Categoría	Coste (sin IVA)	% IVA	Coste (con IVA)
Ordenador de sobremesa	1.138,78€	18	1.343,76€
Ordenador portátil	507,63€	18	599,00€
Samsung Galaxy S3	422,88€	18	499,00€
Samsung Galaxy Mini	90,08€	21	109,00€
Sony Xperia U	162,21€	21	196,27€
Asus Transformer TF101	338,14€	18	399,00€

Tabla 50. Coste de los equipos

En la *Tabla 51* se muestra la amortización de los equipos:

Categoría	Coste	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Ordenador de sobremesa	1.138,78€	100	8	60	151,84€
Ordenador portátil	507,63€	100	8	60	67,68€
Samsung Galaxy S3	422,88€	100	1	60	7,05€
Samsung Galaxy Mini	90,08€	100	1	60	1,50€
Sony Xperia U	162,21€	100	1	60	2,70€
Asus Transformer TF101	338,14€	100	1	60	5,64€
Total: 2.659,72€					**Total: 236,41€**

Tabla 51. Amortización de los equipos

Para el cálculo de la amortización se utiliza la siguiente fórmula:

$$Coste\ Imputable = \frac{A * B * C}{C}\ (Euros)$$

Siendo:

- A: nº de meses desde la fecha de facturación en que el equipo es utilizado
- B: periodo de depreciación (60 meses)
- C: coste del equipo (sin IVA)
- D: Porcentaje del uso que se dedica al proyecto (habitualmente 100%)

Detallar también los gastos correspondientes a las licencias utilizadas para el desarrollo, mostrados en la *Tabla 52*:

Descripción	Empresa	Costes imputable
Licencia Microsoft Office Proffesional 2013 [18]	Microsoft	539,00€
Licencia Microsoft Project Proffesional 2013 [19]	Microsoft	1.369,00€
		Total: 1.908,00€

Tabla 52. Coste de las licencias de los programas

Una vez sumados los gastos para el proyecto se obtiene el total mostrado en la *Tabla 53*, al que hay que sumarle un 20% de los costes indirectos.

Descripción Costes	Costes Totales
Personal	17.808,12€
Amortización	236,41€
Costes de licencias	1.908€
Costes Indirectos (20%)	3.990,51€
Total	**23.943,04€**

Tabla 53. Costes totales

Finalmente al total de los gastos del proyecto hay que sumarle un 21% correspondiente al IVA que conlleva la realización de éste. En la *Tabla 54* se muestra por tanto el coste bruto del proyecto.

Descripción Costes	Costes Totales
Coste sin IVA	23.943,04€
IVA	21%
Total	**28.971,08€**

Tabla 54. Presupuesto total con IVA

7.3 CONCLUSIÓN DEL CAPÍTULO

El coste total del proyecto asciende a una suma de 28.971,08 Euros. Esta inversión se puede recuperar a través de publicidad en la propia aplicación o mediante su venta en la tienda de aplicaciones en línea de Google denominada Google Play.

- Para recuperar el 100% de la inversión mediante la venta de la aplicación a un precio de 1.50 euros, se necesitaría vender un total de 28990 aplicaciones, dado que para poder poner una aplicación en venta hay que estar dado de alta como desarrollador y pagar 19,25€ anuales, correspondientes a los 25 USD (United States Dollar [25]) que establece Google [20] (1 USD=0.77€ [21]) y de lo que se obtiene de cada venta Google retiene un 30% del valor de la aplicación [22]. El cálculo del numero de aplicaciones sigue la siguiente fórmula:

$$nº\ apps = \frac{Coste\ total\ proyecto + Coste\ alta\ desarrollador}{70\ \%\ coste\ total\ aplicacion\ en\ Google\ Play}$$

$$nº\ apps = \frac{28.971{,}08 + \ 19{,}25}{1{,}05} = 28.989{,}41 = 28990\ aplicaciones$$

- La opción de recuperar el dinero mediante el visionado de publicidad dentro de la aplicación implica un desconocimiento del momento en que el pay-back se hace positivo, es decir, el periodo de tiempo que se necesita para recuperar la inversión. Esto se debe a que este método depende de muchos factores como son los accesos a la aplicación, el dinero obtenido por cada visionado etc.

Por lo tanto la mejor opción para la recuperación de la totalidad de la inversión es poner la aplicación en venta en el Google Play e introducir publicidad en la misma.

CAPÍTULO 8. BIBLIOGRAFÍA

1. Android (Consultado el: 16/04/2013)

https://es.wikipedia.org/wiki/Android

2. IOS (Consultado el: 16/04/2013)

http://es.wikipedia.org/wiki/IOS_(sistema_operativo)

3. App Store (Consultado el: 16/04/2013)

http://es.wikipedia.org/wiki/App_Store

4. Google Play (Consultado el: 16/04/2013)

http://es.wikipedia.org/wiki/Google_Play

5. Activity (Consultado el: 17/04/2013)

http://developer.android.com/reference/android/app/Activity.html

6. XML (Consultado el: 18/04/2013)

http://es.wikipedia.org/wiki/Extensible_Markup_Language

7. JSON (Consultado el: 18/04/2013)

http://es.wikipedia.org/wiki/Json

8. Servidor (Consultado el: 22/04/2013)

http://es.wikipedia.org/wiki/Servidor

9. Interfaz de entrada común (Consultado el: 23/04/2013)

http://es.wikipedia.org/wiki/Interfaz_de_entrada_com%C3%BAn

10. Servlet (Consultado el: 24/04/2013)

http://es.wikipedia.org/wiki/Servlet

11. Base de datos (Consultado el: 29/04/2013)

http://es.wikipedia.org/wiki/Base_de_datos

12. Base de datos relacional (Consultado el: 30/04/2013)

http://es.wikipedia.org/wiki/Base_de_datos_relacional

13. MySQL (Consultado el: 30/04/2013)

http://es.wikipedia.org/wiki/Mysql

14. Base de datos documental (Consultado el: 06/05/2013)

http://es.wikipedia.org/wiki/Base_de_datos_documental

15. MongoDB (Consultado el: 06/05/2013)

http://es.wikipedia.org/wiki/MongoDB

16. Guía presupuestaria (Consultado el: 27/05/2013)

https://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20%283%29_1.xlsx

17. Sueldo verificador de pruebas (Consultado el: 27/05/2013)

http://www.expoqa.com/pdf/expoqa12/SalariosTestersEspana-ES-V03.pdf

18. Coste licencia office (Consultado el: 28/05/2013)

http://www.microsoftstore.com/store/mseea/es_ES/cat/ThemeID.30303100/categoryID.61511100?WT.mc_id=MSCOM_ES_ES_HP_Shop_Office

19. Coste licencia Project (Consultado el: 28/05/2013)

http://www.microsoftstore.com/store/mseea/es_ES/pdp/productID.263156000

20. Registro desarrolladores (Consultado el: 29/05/2013)

https://support.google.com/googleplay/android-developer/answer/113468?hl=es&ref_topic=2897388

21. Conversor de moneda (Consultado el: 29/05/2013)

http://www.xe.com/es/

22. Tarifa de transacción (Consultado el: 29/05/2013)

https://support.google.com/googleplay/android-developer/answer/112622?hl=es&ref_topic=2897388

23. Consultas de problemas de implementación de código de la aplicación (Consultado durante el periodo de desarrollo [17/10/2013 - 08/03/2013])

http://developer.android.com/reference/packages.html

http://stackoverflow.com/

A continuación se muestran los enlaces de algunas imágenes utilizadas a lo largo de la memoria

24. Imagen de base de datos (Consultado el: 16/04/2013)

http://www.coloralcuadrado.com/blog/wp-content/uploads/2011/10/Database_1-300x300.png

25. Imagen de servidor (Consultado el: 16/04/2013)

http://www.atjeu.com/wp-content/uploads/2012/06/servers.png

26. Imagen de móvil (Consultado el: 16/04/2013)

http://img.xataka.com.mx/2012/04/galaxy-i9300-userguide1.jpg